Universität Klagenfurt

- Institut für Informatiksysteme -

Univ.-Prof. Dipl.-Ing. Dr. Martin Hitz

UNTERSUCHUNG UND VERGLEICH UML-BASIERTER ENTWICKLUNGSWERKZEUGE

Kulmesch Markus	**Schelander Alexander**	**Türk Adolf**
Informationsmanagement Matrikel-Nr. 9960373 Obersammelsdorf 41, 9122 St. Kanzian	Informationsmanagement Matrikel-Nr. 0360209 Unterbergen 2, 9321 Kappel/Kr.	Informationsmanagement Matrikel-Nr. 0160329 Enzenbergstrasse 25, 9020 Klagenfurt
1. Version	2. Version	2. Version

Inhaltsverzeichnis

1 Einleitung 1
1.1 Aufbau 1
1.2 Testbedingungen 1
2 Werkzeugliste 2
2.1 Einsatzbereich 2
2.2 Grobauswahl-Kriterien 2
2.3 Grobauswahl-Werkzeugliste 3
2.3.1. Poseidon for UML 3
2.3.2. StarUML 4
2.3.3. Visual UML 4
2.3.4. Altova Umodel 5
2.3.5. Objecteering 5
2.3.6. Visual Paradigm for UML 6
2.3.7. Magic Draw 7
2.3.8. ObjectIF 8
2.4 Ausgewählte Werkzeuge 10
2.4.1. StarUML 11
2.4.2. Poseidon for UML 13
2.4.3. Objecteering 14
2.4.4. Visual Paradigm for UML 15
2.4.5. Magic Draw 16
2.4.6. ObjectIF 17
3 UML-Konformität in der Darstellung 18
4 Kriterienkatalog 22
4.1 Methodenbasierte Kriterien 22
4.2 Funktionale Kriterien 24
4.3 Ergonomie 29
5 Evaluationen 33

5.1 StarUML 33
5.1.1. Methodenbasierte Kriterien 33
5.1.2. Funktionale Kriterien 35
5.1.3. Ergonomie 39
5.2 Poseidon for UML 42
5.2.1. Methodenbasierte Kriterien 42
5.2.2. Funktionale Kriterien 43
5.2.3. Ergonomie 47
5.3 Objecteering 6.1 Free Edition 50
5.3.1. Methodenbasierte Kriterien 50
5.3.2. Funktionale Kriterien 52
5.3.3. Ergonomie 55
5.4 Visual Paradigm for UML 6.4 58
5.4.1. Methodenbasierte Kriterien 58
5.4.2. Funktionale Kriterien 60
5.4.3. Ergonomie 64
5.5 Magic Draw 67
5.5.1. Methodenbasierte Kriterien 67
5.5.2. Funktionale Kriterien 69
5.5.3. Ergonomie 73
5.6 ObjectIF 76
5.6.1. Methodenbasierte Kriterien 76
5.6.2. Funktionale Kriterien 78
5.6.3. Ergonomie 81
6 Empirische Bewertung der Werkzeug-Bedienung 85
6.1 GOMS 85
6.2 Beispielprojekt: Onlinebuchgeschäft 88
6.4 Erstellung des Klassendiagramms 89
6.4.1. Zusammenfassung der Ergebnisse 89
6.4.2. Interpretation der Ergebnisse 89

6.4.3. Klassendiagramm – StarUML 91
6.4.4. Klassendiagramm - Poseidon for UML 94
6.4.5. Klassendiagramm – Objecteering 98
6.4.6. Klassendiagramm – Visual Paradigm 102
6.4.7. Klassendiagramm - Magic Draw 106
6.4.8. Klassendiagramm – ObjectIF 110
6.5 Erstellung des Anwendungsfalldiagramms 114
6.5.1. Zusammenfassung der Ergebnisse 114
6.5.2. Interpretation der Ergebnisse 114
6.5.3. Anwendungsfalldiagramm – StarUML 115
6.5.4. Anwendungsfalldiagramm – Poseidon for UML 117
6.5.5. Anwendungsfalldiagramm – Objecteering 119
6.5.6. Anwendungsfalldiagramm – Visual Paradigm 121
6.5.7. Anwendungsfalldiagramm – Magic Draw 123
6.5.8. Anwendungsfalldiagramm – ObjectIF 125
6.6 Erstellung des Aktivitätsdiagramms 127
6.6.1. Zusammenfassung der Ergebnisse 127
6.6.2. Interpretation der Ergebnisse 127
6.6.3. Aktivitätsdiagramm – StarUML 128
6.6.4. Aktivitätsdiagramm – Poseidon for UML 130
6.6.5. Aktivitätsdiagramm – Objecteering 132
6.6.6. Aktivitätsdiagramm – Visual Paradigm 134
6.6.7. Aktivitätsdiagramm – Magic Draw 136
6.6.8. Aktivitätsdiagramm – ObjectIF 138
6.7 Erstellung des Komponentendiagramms 140
6.7.1. Zusammenfassung der Ergebnisse: 140
6.7.2. Interpretation der Ergebnisse 140
6.7.3. Komponentendiagramm – StarUML 141
6.7.4. Komponentendiagramm – Poseidon for UML 143
6.7.5. Komponentendiagramm – Objecteering 145

6.7.6. Komponentendiagramm – Visual Paradigm 147
6.7.7. Komponentendiagramm – Magic Draw 149
6.8 Erstellung des Zustandsdiagramms 151
6.8.1. Zusammenfassung der Ergebnisse 151
6.8.2. Interpretation der Ergebnisse 151
6.8.3. Zustandsdiagramm – StarUML 152
6.8.4. Zustandsdiagramm – Poseidon for UML 154
6.8.5. Zustandsdiagramm – Objecteering 156
6.8.6. Zustandsdiagramm – Visual Paradigm 158
6.8.7. Zustandsdiagramm – Magic Draw 160
6.8.8. Zustandsdiagramm – ObjectIF 162
6.9 Erstellung des Verteilungsdiagramms 166
6.9.1. Zusammenfassung der Ergebnisse 166
6.9.2. Interpretation der Ergebnisse 166
6.9.3. Verteilungsdiagramm – StarUML 167
6.9.4. Verteilungsdiagramm – Poseidon for UML 169
6.9.5. Verteilungsdiagramm – Objecteering 171
6.9.6. Verteilungsdiagramm – Visual Paradigm 173
6.9.7. Verteilungsdiagramm – Magic Draw 175
7 Gegenüberstellung der UML Werkzeuge 177
8 Zusammenfassung 181
9 Anhang 185
9.1 Kriterienkatalog aus Kapitel 4 185
10 Literaturverzeichnis 193

Zusammenfassung: „UNTERSUCHUNG UND VERGLEICH UML-BASIERTER ENTWICKLUNGSWERKZEUGE“

Von März 2007 bis März 2009 wurden die neuesten Versionen der sechs UML-basierten Entwicklungswerkzeuge StarUML, Poseidon for UML, Objecteering, Visual Paradigm for UML, MagicDraw und ObjectIF auf ihre Einsatztauglichkeit hin untersucht.

Es gibt keine großen Unterschiede bei den methodenbasierten Kriterien zwischen den sechs Tools. Mit allen Werkzeugen lassen sich die verschiedenen Diagramme einfach anlegen. Modellierungsinformationen werden zentral in einem Repository gespeichert. OCL wird von Objecteering, Visual Paradigm und Magic Draw unterstützt. Einzig ObjectIF ist in der Anwendung wenig intuitiv. Auch ist es mit diesem Tool schwer mögliche Zusammenhänge zwischen verschiedenen Diagrammtypen herzustellen. Mit Ausnahme von ObjectIF kann auch das Abstraktionsniveau eingestellt werden. Am besten schneidet für diesen Bereich Magic Draw ab, gefolgt von Visual Paradigm.

Auch bei der funktionalen Bewertung hat Magic Draw die Nase vorn. Die leichte und übersichtliche Navigation sticht besonders hervor. Mit Magic Draw ist es einfach möglich, automatisch Reports zu generieren und Diagramme in Datenbankschemas zu transformieren. Visual Paradigm verfügt über die meisten Funktionalitäten dieses Bereichs und ist auch von der Navigation annähernd gleich wie Magic Draw.

Gefolgt werden die zwei Spitzenreiter von StarUML, Poseidon for UML, Objecteering und ObjectIF. Diese vier Tools schneiden in diesem Bewertungsbereich fast gleich ab. StarUML ist gut in der Navigation und in den Browsing-Eigenschaften. Poseidon for UML enthält keine Konsistenz- bzw. Integritätsprüfung. Diese vier Tools schneiden bei der Report-Ausgabe schlecht ab. ObjectIF punktet wiederum bei der Diagrammausgabe und ermöglicht wie Visual Paradigm und Magic Draw eine Mehrbenutzerverwaltung mittels Server.

Bei der Ergonomie schneiden StarUML, Visual Paradigm und Magic Draw am besten ab. Sie verfügen über eine integrierte und ausführliche Hilfefunktion. Auch Poseidon for UML besitzt eine Hilfe, jedoch nur im HTML-Format. Eine Suche nach Themengebieten ist nicht möglich.

Objecteering ist beim Anlegen von Diagrammen leicht verwirrend. Die Bedienung von Poseidon for UML ist gewöhnungsbedürftig. ObjectIF ist durch die Verwaltung mittels eigenständiger Fenster schwer überschaubar. Die Handhabung des Modelleditors hingegen ist leicht verständlich. Außerdem bietet ObjectIF eine Online-Hilfe.

Abhängig von den gebotenen Funktionalitäten kann es bei der Performance zu leichten Verzögerungen kommen. Performanceprobleme sind z.B. beim Testen von Poseidon for UML“ einige Male aufgetreten. Hat der Benutzer mehrere Diagramme auf einmal geöffnet, können geringe Wartezeiten auf ein Systemfeedback auftreten. Bei der Performance schneiden StarUML und Objecteering am besten ab. Dies ist wahrscheinlich darauf zurück zu führen, dass Visual Paradigm und Magic Draw vom Funktionsumfang vielleicht etwas überladen sind.

Zusammenfassend sind Magic Draw und Visual Paradigm die Sieger der Bewertung. Mit diesen Tools werden alle wichtigen Funktionalitäten für die UML-Modellierung, das automatische Generieren von Code und das Reverse Engineering geliefert. Der Umgang mit den Tools ist leicht erlernbar und intuitiv. ObjectIF ist wegen der mangelnden Benutzerfreundlichkeit und Anpassungsmöglichkeiten nicht zu empfehlen.

Schelander Alexander	**Türk Adolf**
Informationsmanagement Matrikel-Nr. 0360209 Unterbergen 2, 9321 Kappel/Kr.	Informationsmanagement Matrikel-Nr. 0160329 Enzenbergstrasse 25, 9020 Klagenfurt
2. Version	2. Version

1 Einleitung

Diese Arbeit wurde im Rahmen des 8-Stunden Softwarepraktikums verfasst. Ziel dieser Arbeit ist es die Leistungsfähigkeit aktueller UML-Modellierungswerkzeuge zu untersuchen.

1.1 Aufbau

Diese Arbeit gliedert sich in folgende Abschnitte:

- In Kapitel 2 wird der Auswahlprozess der UML-Werkzeuge beschrieben. Es werden die Kriterien für die Grobauswahl angeführt. Das Ergebnis ist eine Werkzeugliste mit acht Produkten. Sechs dieser acht Produkte (StarUML, Poseidon for UML, Objecteering, Visual Paradigm, Magic Draw und ObjektIF) werden im Weiteren näher analysiert.
- Basis für die nähere Bewertung der Werkzeuge ist der Kriterienkatalog, der im vierten Abschnitt dieser Arbeit festgelegt wurde. Anhand dieser Kriterien erfolgt die Analyse der UML-Werkzeuge.
- Das Kapitel 3 beinhaltet die Resultate der Untersuchung der sechs UML-Werkzeuge bezüglich deren Konformität in der UML-Darstellung.
- Die empirische Bewertung ist ein weiterer Punkt. Anhand von verschiedenen UML-Diagrammen werden die sechs UML-Produkte bezüglich der Bedienbarkeit und der Effizienz der Benutzung gestestet. Außerdem wird auf GOMS detaillierter eingegangen.

1.2 Testbedingungen

Die ausgewählten Softwareprodukte wurden auf einem tragbaren Rechner (Laptop) mit dem Betriebssystem Windows XP – SP 3 getestet. Unter dem 1,79 GHz Prozessor und 512 MB RAM Arbeitsspeicher konnte mit allen Produkten flüssig gearbeitet werden, d.h. die Rechenleistung war für die Durchführung der Evaluation ausreichend.

Einige Produkte wurden von den Herstellern nur als Demoversionen zur Verfügung gestellt. Diese waren zeitlich (meistens 30-tage Testversionen) oder funktionell in ihrem Umfang eingeschränkt.

Diese Evaluierung ist natürlich nur eine Momentaufnahme (März 2007 bis März 2009) der aktuellen Situation. Schon während der Evaluationsphase veröffentlichten einige Hersteller neue Versionen ihres Produktes. Im Rahmen dieser Arbeit wurde versucht, diese Änderungen zu laufend zu berücksichtigen.

2 Werkzeugliste

In der ersten Auswahlphase wurden acht Produkte anhand eines groben Kriterienkatalogs näher betrachtet. Die Produktrecherche wurde größtenteils über das Internet durchgeführt. Ausgangspunkte für die Suche waren aber auch UML-Modellierungswerkzeuge aus bereits absolvierten Lehrveranstaltungen, wie z.B. StarUML aus der Lehrveranstaltung „Objektorientierte Programmierung“.

2.1 Einsatzbereich

Die untersuchten Modellierungswerkzeuge sollten für den Einsatz in verschiedenen Softwarepraktika an der Universität geeignet sein, wo meisten 2-4 Personen an einem Projekt beteiligt sind. Weiters sollten die Werkzeuge den UML 2.0 Standard unterstützen. Auf Basis dieser Vorgaben und den folgenden Kriterien wurde eine Grobauswahl durchgeführt.

2.2 Grobauswahl-Kriterien[1]

- Die UML-Werkzeuge sollten kostenlos verfügbar, oder zumindest gratis testbar sein.
- Kleine Software-Projekte (2-4 Personen) sollen unterstützt werden.
- Die Produkte sollen auf einem Windows-Betriebssystem lauffähig sein.
- Der Umgang mit den Produkten soll schnell erlernbar sein, da lange Einarbeitungsphasen aus zeitlichen Gründen unvorteilhaft wären.
- Weiters sollen für die selbstständige Arbeit mit den Werkzeugen Tutorials, Dokumentationen, weiterführende Links bzw. ähnliche Hilfestellungen zur Verfügung stehen.
- Support: Bietet der Hersteller irgendwelche Formen von Anwenderunterstützung an?
- Eine automatische Codegenerierung für die an der Universität gängigsten Programmiersprachen (Java, C#, C++) soll unterstützt werden.
- Die Bedienung soll einfach und ähnlich anderen Entwicklungswerkzeugen sein.
- Weiters sollen die Produkte eine Zusammenarbeit mit anderen Entwicklungswerkzeugen, wie Programmierumgebungen (z.B. Eclipse, JBuilder, usw.) unterstützen.
- Der Standardmodellierungssprache ist UML 2.0.
- Ist der Hersteller einer der „führenden“ Anbieter, und ist es wahrscheinlich, dass er in einigen Jahren noch existiert?

[1] Vgl. U. Kelter, Bewertung und Auswahl von UML-Werkzeugen, 2001

- Es soll möglich sein die gängigsten Diagrammarten modellieren zu können.
- Eine automatische Konsistenzprüfung bezüglich der Integrität eines Modells ist von Vorteil.

2.3 Grobauswahl-Werkzeugliste

Durch eine Internetrecherche ergab sich eine Werkzeugliste mit acht UML-Werkzeugen:

- Poseidon for UML
- StarUML
- Visual UML
- Altova Umodel
- Objecteering
- Visual Paradigm
- Magic Draw
- ObjectIF

Im Folgenden werden alle oben angeführten Produkte näher beschrieben.

2.3.1. Poseidon for UML

Poseidon for UML ist eine Weiterentwicklung des Open-Source Projektes ArgoUML. Neben der im vollen Umfang, aber zeitlich eingeschränkt, nutzbaren Community Edition existieren auch kommerzielle Varianten (Standard, Professional, Embedded Edition) für die Support und Schulung angeboten werden. Der Download erfolgt unter www.gentleware.com, benötigt Java JRE oder JDK.

Da Poseidon for UML in der Programmiersprache Java geschrieben ist, ist dieses Produkt plattformunabhängig. Die Hardwaremindestanforderungen sind ein Pentium III Prozessor und mindestens 512 MB Arbeitsspeicher.

Die Größe der Downloaddatei beläuft sich auf nur 23 MB. Der Benutzer findet genügend Onlineinformationen, die aber nur in englischer Sprache vorliegen. Die Programmsprache ist aber Deutsch. Die Arbeitsumgebung ist übersichtlich aufgebaut. Direkt auf der Herstellerseite findet man auch Tutorials mit Beispielprojekten, kostenlosen UML-Onlinekursen, Einführungen in Poseidon und Einführungen in die objektorientierte Programmierung. Weiterführende Hilfe bieten diverse Onlineforen, ein Benutzerhandbuch ist ebenfalls kostenlos erhältlich. Der technische Support und Schulungen, die angeboten werden, sind kostenpflichtig.

Poseidon for UML unterstützt folgende Diagrammtypen: Anwendungsfalldiagramme, Klassendiagramme, Zustandsdiagramme, Aktivitätsdiagramme, Sequenzdiagramme, Kollaborationsdiagramme, Komponentendiagramme und Verteilungsdiagramme.

2.3.2. StarUML

Hierbei handelt es sich um eine Open-Source UML/MDA Plattform. StarUML kann unter staruml.sourceforge.net herunter geladen werden.

Diese Software ist nur unter dem Betriebssystem Windows 2000 oder XP lauffähig. Die Mindestanforderungen belaufen sich auf 233 MHz Rechenleistung, 128 MB RAM sowie 110 MB an freier Festplattenkapazität. Das es sich hierbei um eine Open-Source Software handelt, fallen keine Kosten für Download und Benutzung an. Der Support ist jedoch kostenpflichtig. Die Dokumentationen für Anwender und Entwickler (API Informationen) sind online verfügbar. Ebenso ist ein Forum für etwaige Fragen eingerichtet. Die Open-Source Gemeinde treibt die Weiterentwicklung dieses Produktes voran.

Die Benutzersprache ist Englisch. Es existieren auch Erweiterungen, wie ER-Extensions, die integriert werden können. StarUML unterstützt UML-Werkzeug UML 2.0. Eine Codegenerierung für die gängigsten Programmiersprachen ist ebenso enthalten. Mit StarUML kann man folgende Diagramme umsetzen: Klassendiagramme, Anwendungsfalldiagramme, Sequenzdiagramme, Kollaborationsdiagramme, Zustandsdiagramme, Aktivitätsdiagramme, Komponentendiagramme, Kommunikationsdiagramme und Strukturdiagramme.

2.3.3. Visual UML

Visual UML kann über das Internet unter der Adresse www.visualuml.com bezogen werden. Eine vollfunktionale 30-Tage Demoversion ist als Download verfügbar. Weiters existieren neben der Standard Edition noch die Developer Edition und die Plus-Developer Edition (mit VBA).

Visual UML ist schon ab Windows 95 lauffähig und benötigt nur 50 bis 70 MB Speicherplatz auf der Festplatte. Die aktuelle Version ist 5.3.

Die Benutzersprache ist Englisch. Die Anordnung der Programmelemente und die Benutzung sind MS Office-Programmen nachempfunden. Jedoch Besitzt Visual UML viele gewöhnungsbedürftige Icons. Dieses Tool verfügt über ein Onlinetutorial mit Einführungshilfen und Beispielprojekten. Mit diesem Produkt können Anwendungsfalldiagramme, Klassendiagramme, Zustandsdiagramme, Aktivitätsdiagramme, Sequenzdiagramme, Kollaborationsdiagramme, Komponentendiagramme und Verteilungsdiagramme erzeugt werden.

2.3.4. Altova Umodel

Die 30-tägige, voll funktionsfähige Testversion des UML-Werkzeugs Altova Umodel erreicht man über die URL der Herstellers, www.altova.com. Diese Software ist nur auf Windows Rechnern installierbar (Windows 2000, NT oder XP).

Um flüssig arbeiten zu können, werden mindestens 512 MB Arbeitsspeicher benötigt. Der Benutzer kann zwischen Englisch oder Deutsch als Benutzersprache wählen. Die einzelnen Bildschirminhalte sind übersichtlich angeordnet und erinnern an ähnliche Entwicklungswerkzeuge. Weiters sind Tutorials und Beispielprojekte vorhanden. Hilfe zum Umgang mit Altova Umodel kann über die Funktionstaste F1 aufgerufen werden.

Das UML 2.0 Werkzeug enthält auch eine automatische Codegenerierung für Java und C#. Praktische Features sind auch das Reverse Engineering, die Modellanalyse und die automatische Syntaxprüfung. Mit Altova Umodel kann man die gängigsten Diagrammarten erstellen. Das wären: Anwendungsfalldiagramme, Klassendiagramme, Zustandsdiagramme, Aktivitätsdiagramme, Sequenzdiagramme, Kollaborationsdiagramme, Komponentendiagramme und Verteilungsdiagramme.

2.3.5. Objecteering

Das Programm kann über die englischsprachige Homepage http://www.objecteering.com/ heruntergeladen werden. Er werden eine Free Edition, die unbegrenzt lauffähig ist und eine 30-Tage-Testversion der Enterprise Edition angeboten.

Beide Versionen sind für Windows 2000/XP und für Linux RedHat 9.0 & Enterprise 4.0 erhältlich.

Die Enterprise Edition kostet inklusive Dokumentationsproduktion 990,-- €.

Die Free Edition unterscheidet sich von der kommerziellen Version im Wesentlichen durch folgende Punkte:

- Es ist keine Importfunktionalität verfügbar
- Die Diagramme enthalten ein Wasserzeichen
- Es kann nur eine einzige MDA Komponente gleichzeitig verteilt werden. MDA steht hierbei für „Model Driven Architecture“, einen modellgesteuerten Ansatz zur Entwicklungsunterstützung. Dieser Ansatz wurde von der OMG (Object Management Group) standardisiert.
- Wenn das Modell 50 Elemente überschreitet, wird die MDA-Komponente vom Projekt abgewählt und kann erst dann wieder angewählt werden, wenn die Anzahl der Elemente wieder unter 50 liegt. Elemente, die in die Kalkulation der 50 Elemente miteinbezogen werden, sind Packages, Classifier, Kollaborationen, Artefakte, Knoten, Klassen, Datentypen, Enumerationen, Signale, Use Cases, Akteure, Interfaces, Template-Parameter und Komponenten.

- Multi-user-, model diff/merge- und CMS-Features werden nicht unterstützt.
- Code-Gernerierung für Corba IDL, SQL DDL und Fortran werden in der Free Edition nicht unterstützt.

Für dieses Projekt wurde die Free Edition evaluiert, weil sie ausreichend ausgestattet ist und weil sie gratis und unbegrenzt lauffähig ist.

Schlüssel-Vorteile von Objecteering:

- Vollständige Abdeckung der modellgetriebenen Entwicklung, von den Anforderungen über Analyse und Design bis hin zum Code und Tests
- Nahtloser Übergang von einer Phase in die nächste, mit voller Rückverfolgbarkeit zwischen Anforderungen, Modellelementen und Code
- Dynamische Vollständigkeits- und Konsistenzprüfung
- Mächtige MDA-Automation, die aus einer verbesserten Modelltransformation und Codegenerierung resultiert
- Integriertes Repository, das sich über alle Entwicklungsphasen zieht
- Automatisierte Dokumentationsgenerierung und Web Publishing
- Integration und Synchronisation mit der Eclipse-Entwicklungsumgebung

Objecteering unterstützt Klassendiagramme, Objektdiagramme, Verteilungsdiagramm, Sequenzdiagramme, Anwendungsfalldiagramme, Zustandsdiagramme, Paketdiagramme, Komponentendiagramme und Aktivitätsdiagramme.

2.3.6. Visual Paradigm for UML

Visual Paradigm for UML ist ein preisgekröntes Produkt von Visual Paradigm International. Es hebt sich von den anderen im Rahmen dieses Projekts vor allem durch seine ausgezeichnete Ausstattung ab. Je nach Version wird zum Beispiel Code-Generierung und Reverse-Engineering für über 10 Programmiersprachen unterstützt.

Die Preise für Visual Paradigm for UML belaufen sich bei der Single Seat License auf 59,-- bis 1.679,-- € und bei der Floating License auf 129,-- bis 2.182,-- €.

Für unser Projekt wurde eine 30-Tage-Testversion der Standard Edition evaluiert.

Für dieses Programm kann sich eine "tertiäre Bildungseinrichtung" als Academic Partner anmelden. Für den Partnertyp "Basic Academic Partner" ist eine institutionsweite Lizenz für die sehr gut ausgestattete Standard Edition kostenlos erhältlich. Für den Einsatz in Universitäts-LVs ist dieses Programm wegen seiner Ausstattung und der Möglichkeit einer akademischen Lizenz sehr zu empfehlen.
Die "Visual Paradigm Standard Edition" ist besser ausgestattet als "Altova UModel" und auch bedienungsfreundlicher. Daher wurde sie bei der Evaluierung Altova UModel vorgezogen.

Die Informationen zur Academic License findet man auf folgenden Websites:

- http://www.visual-paradigm.com/partner/academic/
- http://www.visual-paradigm.com/partner/academic/academicpartnerstypes.jsp

Die Anmeldung als Academic Partner kann unter http://www.visual-paradigm.com/partner/academic/joinacademicprogramhowto.jsp durchgeführt werden.

Der Hersteller gibt folgende Systemanforderungen an:

- Intel Pentium III Prozessor mit 1.0 GHz oder mehr
- Mindestens 256MB RAM, aber 1.0 GB wird empfohlen
- Mindestens 400MB Harddisk
- Microsoft Windows (98, 2000, XP, or Vista), Linux, Mac OS X, Solaris oder alle anderen Java-fähigen Plattformen
- Microsoft Visio 2003 and Microsoft .NET Framework 1.1 werden benötigt um Visio Shape und Import Visio Stencils zu verwenden.

Neben der UML-Modellierung kann Visual Paradigm for UML auch für Datenbankmodellierung, objekt-relationales Mapping, Interoperabilität mit anderen Programmen, IDE-Integration, Anforderungsmodellierung, Geschäftsprozessmodellierung, Teamarbeit, Code-Engineering und Dokumentationsgenerierung verwendet werden.

Visual Paradigm for UML unterstützt Klassendiagramme, Objektdiagramme, Paketdiagramme, Komponentendiagramme, Kompositionsstrukturdiagramme, Verteilungsdiagramme, Anwendungsfalldiagramme, Aktivitätsdiagramme, Zustandsdiagramme, Sequenzdiagramme, Kommunikationsdiagramme und Zeitdiagramme.

Zusätzlich zu diesen Diagrammen unterstützt Visual Paradigm for UML noch einige andere Diagrammarten zum Beispiel für die Datenbankmodellierung.

2.3.7. Magic Draw

Magic Draw, Gewinner verschiedener Auszeichnungen, ist ein Produkt der Firma No Magic, Inc.. Das Tool kann von der englischsprachigen Hompage www.magicdraw.com heruntergeladen werden. Abhängig von der unterstützten Programmiersprache bietet der Hersteller eine Vielzahl verschiedener Editionen dieses Produktes an.

Die Preise für Magic Draw belaufen sich bei der Standalone License auf 125,-- bis 1.233,-- € und bei der Floating License auf 679,-- bis 2.159,-- €.

In dieser Arbeit wurde die 150 Tage Testversion der Enterprise Edition verwendet. Diese Version ist zwar in der Anzahl der Modellierungskomponenten beschränkt, jedoch für die folgende Evaluierung ausreichend.

Ähnlich wie bei Visual Paradigm for UML verfolgt auch Magic Draw ein akademisches Programm. Bildungseinrichtungen die an diesem Programm teilnehmen sind berechtigt Magic Draw Personal Edition ohne Gebühren an ihrem Standort zu benutzen. Genauere Informationen zu diesem Programm findet man auf der Homepage des Herstellers unter der Rubrik „Academic Program“.

Der Hersteller gibt folgende Systemanforderungen an:

- Intel Pentium III Prozessor mit 800 MHz oder mehr
- Mindestens 768 MB RAM, aber 1.0 GB wird empfohlen
- Mindestens 400 MB Harddisk
- Auflösung 800 * 600 64k Farben
- Windows NT/2000/XP/Vista, Linux, Mac OS X oder alle anderen Java-fähigen Plattformen

Magic Draw unterstützt Klassendiagramme, Anwendungsfalldiagramme, Kommunikationsdiagramme, Sequenzdiagramme, Zustandsdiagramme, Zustandsmaschinediagramme, Aktivitätsdiagramme, Implementationsdiagramme, Verteilungsdiagramme, Komponentendiagramme, Kompositionsstrukturdiagramme, Objektdiagramme und Zeitdiagramme.

2.3.8. ObjectIF

ObjectIF kann unter der deutschsprachigen Internetadresse des Herstellers www.microtool.de/objectif/ in mehreren Editionen bezogen werden. Die verschiedenen Editionen richten sich an die Abforderungen der Entwickler.

- Enterprise Edition: deckt das gesamte Spektrum von C++ über C# und Visual Basic .NET bis Java ab. Eine Anbindung an Visual Studio und Eclipse ist außerdem möglich.
- Visual Studio .NET Edition: unterstützt insbesondere die Entwicklung von .NET-Anwendungen. Eine automatische Codegenerierung erfolgt für die Programmiersprachen C# und Visual Basic.
- Eclipse Edition: ist für den Einsatz von Java und Eclipse (Open-Source Entwicklungsumgebung) optimiert.

Die Mindesthardwareanforderungen für die Installation (hierbei handelt es sich um die Version 6.1) und Betrieb der Software sind 850 MHz, 512 MB RAM, Windows 2000, XP, MS Word 2000 oder höher und .NET Framework 2.0. Die Installationsgröße ist abhängig von den Installationsoptionen und reicht von 210 MB bis 310 MB.

Die verfügbare Personal Edition ist mit den unten genannten Einschränkungen kostenlos erhältlich. Maximal können fünf Klassendiagramme, fünf Sequenzdiagramme, fünf Paketdiagramme und fünf Anwendungsfalldiagramme erstellt werden. Die er-

stellten Modelle können bei dieser Ausgabe nicht exportiert werden. Die Lizenzen für die anderen Editionen belaufen sich auf € 490.- bis € 1.990.- pro Arbeitsplatzlizenz.

Die Downloadgröße ist von der gewünschten Edition abhängig und liegt zwischen 104 bis 186 MB. Der Support und die Wartung sind kostenpflichtig. Eine Produktvorführung kann sich der Benutzer auch über Online-Filme bzw. Telepräsentationen ansehen.

Die Benutzersprache ist Deutsch, was den Einstieg in das Arbeiten mit der Software erleichtert. Die Bedienung ist gewöhnungsbedürftig, da ObjectIF aus mehreren „Programmfenstern“ besteht und man dadurch schnell die Übersicht verliert. Ansonsten ist der Funktionsumfang eher gering, es wird aber der UML 2.0 Standard unterstützt. Mit ObjectIF kann man Anwendungsfalldiagramme, Klassendiagramme, Zustandsdiagramme, Aktivitätsdiagramme, Sequenzdiagramme und Komponentendiagramme modellieren.

2.4 Ausgewählte Werkzeuge

Visual UML und Altova Umodel werden für die weitere Evaluierung im Rahmen dieser Arbeit nicht berücksichtigt, da in einer ersten Analyse die anderen sechs Produkte im Funktionsumfang und in der Beschreibung ansprechender waren. Visual UML ist durch seine Icons in der Handhabung sehr gewöhnungsbedürftig. Altova Umodel war zum Zeitpunkt der Evaluierung nur auf Microsoft Windows installierbar.

Auf Basis der im Abschnitt 2.2 angeführten Kriterien und Anforderungen wurden die sechs UML-Werkzeuge StarUML, Poseidon for UML, Objecteering, Visual Paradigm, Magic Draw und ObjectIF ausgewählt. Diese Werkzeuge werden in der weiteren Arbeit näher miteinander verglichen. Ein wichtiges Kriterium war dabei, dass die Werkzeuge kostenlos erhältlich sind (zumindest 30 Tage testbar).

Alle Werkzeuge unterstützen den UML 2.0 Standard, verfügen zum Teil über Multi-User- Eigenschaften, haben eine automatische Konsistenzprüfung und können die gängigsten UML-Diagramme abbilden. Weiters besitzen die Werkzeuge die Möglichkeit einer Codegenerierung für die in den Softwarepraktika eingesetzten gängigsten Programmiersprachen Java, C# und C++.

Die folgende Tabelle gibt eine Übersicht der unterstützten Diagrammtypen.

+	vorhanden
-	nicht vorhanden

Diagrammtyp	Star-UML	Poseidon for UML	Objec-teering	Visual Pardigm	Magic Draw	ObjectIF
Klassendiagramm	+	+	+	+	+	+
Objektdiagramm	+	+	+	+	+	+
Paketdiagramm	-	-	-	+	-	+
Komponentendiagramm	+	+	+	+	+	-
Kompositionsstrukturdiagramm	+	+	-	+	+	-
Verteilungsdiagramm	+	+	+	+	+	-
Anwendungsfalldiagramm	+	+	+	+	+	+
Aktivitätsdiagramm	+	+	+	+	+	+
Zustandsdiagramm	+	+	+	+	+	+
Sequenzdiagramm	+	+	+	+	+	+
Kommunikationsdiagramm	+	-	-	+	+	-
Zeitdiagramm	-	-	-	+	+	-

2.4.1. StarUML

Hersteller:	Open Source Community
URL:	www.staruml.com
Unterstützte Plattform:	Microsoft Windows (32-bit-Version)
Getestete Version:	5.0.2.1570
Lizenz:	GLP (kostenlos)

Die erste Version von StarUML wurde 1996 veröffentlicht. Damals hieß dieses Produkt noch Plastic bzw. Agora Plastic und wurde vom koreanischen Unternehmen Plastic Software entwickelt. Anfangs handelte es sich dabei noch um ein recht einfaches Produkt zur Visualisierung von Softwaremodulen und deren Abhängigkeiten. Durch die Weiterentwicklung des Produktes wurde das UML-Werkzeug immer mächtiger. Im Jahre 2005 wurde das Produkt Plastic zu StarUML weiterentwickelt und kostenlos zur Verfügung gestellt.

StarUML wird als Open-Source Projekt vorangetrieben und kann kostenlos unter der Internetadresse www.staruml.com herunter geladen werden. Zeitliche und funktionale Einschränkungen gibt es beim Open-Source Produkt StarUML keine. Dieses Modellierungswerkzeug ist aber nur in englischer Sprache erhältlich und läuft nur unter einem Microsoft Windows Betriebssystem. Die lokale Benutzerhilfe bzw. Dokumentation ist ausführlich und nur in englischer Sprache.

Für die Modellierung von Diagrammen unterstützt StarUML neun Diagrammtypen. Das sind: Klassendiagramme, Komponentendiagramme, Kompositionsstrukturdiagramme, Verteilungsdiagramme, Anwendungsfalldiagramme, Aktivitätsdiagramme, Zustandsdiagramme, Sequenzdiagramme und Kommunikationsdiagramme. Reine Paketdiagramme können nicht erzeugt werden, es ist aber möglich Pakete in Klassendiagrammen zu integrieren.

Neben dem Modellieren von Diagrammen unterstützt StarUML auch die Möglichkeit einer automatischen Verifikation. Dabei wird auf mögliche Modellierungsfehler überprüft. Dafür werden alle Dateien im XML-Standard gespeichert.

Weitere Features sind UML Profile, Forward und Reverse Code Engineering für die Programmiersprachen C++, C# und Java und der Import und Export von Modellen. Der Import und Export von Diagrammen wird durch den XMI-Standard unterstützt. Weiters können Modelle importiert werden, die mit dem Modellierungswerkzeug Rational Rose entwickelt wurden. Auch der Import aus verschiedenen Frameworks, wie MFC (Microsoft Foundation Classes), .NET (Base Class Libraries), J2EE (Java 2 Enterprise Edition) und J2SE (Java 2 Standard Edition) sind möglich.

Dieses Werkzeug bietet auch die Möglichkeit an, mit UML Profilen zu arbeiten. Neben dem UML Standard Profil können vier weitere Profile gewählt werden: C++ Profil, C# Profil, Java Profil oder ein Java Beans Profil. Dadurch kann die Modellierungsumgebung einer bestimmten Programmiersprache angepasst werden und der Entwickler kann auf sprachspezifische Datentypen, Tags und Stereotype zurückgreifen.

Damit mehrere Anwender gleichzeitig mit dem gleichen Modell arbeiten können, ist es möglich, ein UML Modell auf Fragmente aufzuteilen und dadurch mehrere Dateien eines Modells zu erzeugen.

2.4.2. Poseidon for UML

Hersteller:	Gentleware AG
URL:	www.gentleware.com
Unterstützte Plattformen:	Plattformunabhängig
Getestete Version:	Community Edition 5.0.1
Lizenz:	Kommerzielle Software, 30 Tage Testversion

Poseidon for UML ist ein UML-Modellierungswerkzeug, das von dem Unternehmen Gentleware AG vertrieben wird. Eine kostenlose Testversion der Community Edition kann von der Homepage des Herstellers (www.gentleware.com) bezogen werden.

Vom UML Modellierungswerkzeug Poseidon for UML sind mehrere Editionen verfügbar, die sich an die Anforderungen der Benutzer richten. Die hier getestete Community Edition richtet sich an UML-Anfänger und Studenten und ist für kleinere Projekte ausreichend. Es können neun UML-Diagrammtypen dargestellt werden.

Poseidon for UML versucht sich mit einer intuitiven Bedienung gegenüber anderen UML-Werkzeugen zu differenzieren. Damit soll dem Benutzer der Umgang mit dem Softwareprodukt erleichtert werden und der Lernprozess verkürzt werden. Angeboten werden die Benutzersprachen: Englisch, Deutsch, Französisch, Spanisch und Chinesisch.

Poseidon for UML ist in der Programmiersprache Java implementiert und damit plattformunabhängig. Die einzelnen Diagramme können auch exportiert werden. In der Community Edition ist auch eine Codegenerierung für Java möglich. Die Diagramme werden im XMI-Standard gespeichert und können aus diesem Format importiert werden.

Reverse Engineering ist erst ab der Standard Edition möglich. Auch das Importieren von Java Archiven (.jar) ist in der Community Edition nicht möglich.

Der Softwareanbieter Gentleware geizt nicht mit Hilfe und Supportleitungen. Auf der Herstellerwebsite gibt es neben kostenpflichtigen Supportleistungen auch kostenlose Hilfestellungen. In Foren können Probleme mit Poseidon for UML angesprochen werden, es existieren Onlinekurse über UML, sowie genügend Anwendungsbeispiele.

2.4.3. Objecteering

Hersteller:	Objecteering Software
URL:	www.objecteering.com
Unterstützte Plattformen:	Microsoft Windows
Getestete Version:	Objecteering Modeler 6.1
Lizenz:	Free Edition

Bei Objecteering handelt es sich um ein Tool von Objecteering Software, einem Tochterunternehmen von Softeam. Das Flagschiff Produkt des Unternehmens ist Objecteering. Das Tool bietet eine komplette modellgetriebene Softwareentwicklungshilfe. Dabei orientiert sich das Tool an gängige Standards wie UML2, BPMN, BMM und SysML.

Die Verwendung der Software soll es Teams ermöglichen, Anforderungen zu managen, genaue und vollständige Modelle zu erstellen, Reports und Dokumentationen zu erzeugen und Code automatisiert zu generieren. Die Code-Generierung kann in den Sprachen Java, C#, C++, SQL DLL, CORBA IDL und Fortran.

Neben der Standalone Installation (nur für Windows) gibt es auch die Möglichkeit einer Client-Server Installation für Windows und Linux. Alle Clients die sich zu dem Server verbinden haben den gleichen Zugriff auf alle Ressourcen des Objecteering Servers.

Objecteering präsentiert sich im bekannten Windowsstill. Dies ermöglicht eine intuitive Verwendung für Windows Benutzer. Im Diagrammeditor kann der Benutzer mittels Linksklick UML Element aus der Werkzeugliste im Zeichenbereich platzieren. Mit dem UML-Editor können die verschiedenen UML Diagramme angelegt werden.

Das Tool verfügt über eine umfangreiche Dokumentation sowie über User Guides. Weiters werden Tutorials angeboten, die auch direkt über einem Link im Tool online aufgerufen werden können. Dokumentation, User Guides und Tutorials sind auf Englisch.

Angaben zum Support findet man auf der Homepage des Herstellers www.objecteering.com. Hier findet man Dokumentationen, einen Online-Support und die Nummer für die Support Hotline. Zusätzlich besteht die Möglichkeit Training und Consulting in Anspruch zu nehmen.

2.4.4. Visual Paradigm for UML

Hersteller:	Visual Paradigm International
URL:	www.visual-paradigm.com
Unterstützte Plattformen:	Alle Java-fähigen Plattformen
Getestete Version:	Standard Edition 6.4
Lizenz:	Kommerzielle Software, 30 Tage Testversion

Visual Paradigm for UML ist ein Tool zur Modellierung mit der UML und ermöglicht darüber hinaus eine Code-Generierung und Reverse Engineering. Der Hersteller Visual Paradigm International ist ein Unternehmen mit Firmenhauptsitz in Hong Kong und bietet neben Visual Paradigm for UML eine Reihe weitere Produkte für den Softwareentwicklungslebenszyklus. Diese Produkte können gebündelt in der Visual Paradigm Siute erworben werden.

Die Code-Generierung unterstützt die Sprachen Java, C++, C#, CORBA IDL, PHP, XML Schema, Ada, Python, VB .NET, Object Definition Language (ODL), Flash ActionScript, Delphi, Perl, Objective-C, and Ruby. Reverse Engineering unterstützt die Sprachen Java, C++, CORBA IDL, PHP, XML Schema, Ada and Python.

Visual Paradigm for UML zeichnet sich besonders durch die intuitive Handhabung und die gut strukturierte Arbeitsfläche aus. Der Stil und die Funktionalitäten orientieren sich an Microsoft Windows. Der Benutzer sollte auch ohne Kenntnis des Benutzerhandbuches in der Lage sein verschiedene UML-Diagramme zu erstellen und diese zu verwalten.

Ein weiteres hilfreiches Feature ist der Report Generator. Mit ihm ist es möglich HTML und PDF Reports für die einzelnen UML Diagramme zu erstellen.

Das Unternehmen versucht sich beim Support von seinen Mitbewerbern zu differenzieren. Auf der Homepage des Betreibers findet man eine ausführliche Dokumentation der einzelnen Tools. Der Benutzer kann sich ein eigenes Hilfetool lokal downloaden und installieren, welches direkt bei der Arbeit verwendet werden kann.

Visual Paradigm International bietet ein Call Back Service für technische Fragen. Zusätzlich gibt es einen technischen Online-Service, einen Online-Live-Chat, ein Diskussionsforum und einen Sales Support. Benutzerseitig ist es möglich Feature-Anfragen zu stellen und Bugs zu melden und somit Einfluss auf die Entwicklung der einzelnen Tools zu nehmen.

2.4.5. Magic Draw

Hersteller:	No Magic, Inc.
URL:	www.magicdraw.com
Unterstützte Plattformen:	Alle Java-fähigen Plattformen
Getestete Version:	Enterprise Edition 16.0
Lizenz:	Kommerzielle Software, 150 Tage Testversion

Magic Draw ist ein Toll zur Modellierung von Geschäftsprozessen, Architekturen, Software und Systemen mit Teamworkunterstützung. Neben dem Modellierungsbereich verfügt das Tool über eine automatisierte Code-Generierung aus UML Diagrammen und Reverse Engineering aus bestehendem Code.

Die Code-Generierung unterstütz die Sprachen Java, EJB, C#, C++, CORBA IDL, DDL, WSDL und XML Schema. Das Reverse Engineering kann aus den Sprachen Java, C#, C++, CORBA IDL, EJB 2.0, DDL, CIL (MSIL), WSDL, und XML Schema erfolgen.

Das Tool verfügt weiters über die Möglichkeit Reports automatisiert zu erstellen. Die Dokumentationen können anschließend als HTML, PDF oder RTF gespeichert werden.

Die Bedienung des Tools ist sehr intuitiv und ermöglicht das Erstellen von UML Diagrammen ohne spezielle Vorkenntnisse. Der Benutzer kann auch Assistenten für die Erstellung verschiedene Diagramme aufrufen, die einen Einstieg erleichtern. Vereinfacht wird die Handhabung durch die deutschsprachige Oberfläche. Hierbei wäre kritisch anzumerken, dass das Benutzerhandbuch nur auf Englisch zur Verfügung steht.

Das Unternehmen bietet seinen Kunden einen umfangreichen Support für das Tool. Neben einer umfassenden Dokumentation und Toolbeschreibung auf der Hompage des Herstellers, gibt es die Möglichkeit den Online Support zu nutzen, oder seine Fragen in Newsgroups zu diskutieren. Kunden können Fehler über Bug Reports melden und so zur Produktentwicklung beitragen.

Gegen eine Gebühr können Benutzer den Umgang mit dem Tool in speziellen Trainings oder Kursen erlernen. Hierbei gibt so sogar die Möglichkeit ein Online Training zu buchen. Weiters kann man, ebenfalls gegen eine Gebühr, Consulting für verschiedene Bereiche in Anspruch zu nehmen.

2.4.6. ObjectIF

Hersteller:	microTOOL GmbH
URL:	www.microtool.de
Unterstützte Plattformen:	Microsoft Windows
Getestete Version:	Personal Edition 7.0
Lizenz:	Free Edition

ObjectIF repräsentiert ein Tool für eine modellgetriebene Softwareentwicklung mit der UML und BPMN in Java, C#, C++, BPEL, XSD und WSDL. Für die Modellierung mit UML oder BPMN besitzt das Tool ein Modellierungs-Framework. Nach der Erstellung der Modelle können diese anschließend iterativ in technische Modelle des Softwareentwurfes oder in Code transformiert werden.

Die Code-Generierung aus dem UML-Modell erfolgt per Mausklick. Das besondere dabei ist, dass der Code ohne Nachbearbeitung kompilierbar ist. Durch Code-Skripten kann diese Code-Generierung benutzerspezifisch angepasst und erweitert werden.

Der Hersteller bietet abhängig von den Anforderungen der Entwicklung verschiedene Editionen dieser Software zum Download an. Eine kostenlose Version von ObjectIF für Eclipse und ObjectIF für Visual Studio .Net kann von der Homepage des Herstellers (www.microtool.de) erworben werden.

Bedingt durch die Tatsache, dass ObjectIF auch auf eine Code Erstellung abzielt, ist der Einstieg für den Benutzer wenig intuitiv. Für ungeübte Benutzer ist es notwendig im Benutzerhandbuch nachzulesen, um den Modellierungseditor zu finden bzw. verwenden zu können. Diesbezüglich ist zu erwähnen, dass ObjectIF über eine gute Dokumentation verfügt, verbunden mit einer Suche und Tutorials. Diese Hilfemaßnahmen bieten eine gute Grundlage für die Verwendung und den Einsatz des Tools.

Support findet man ebenfalls auf der Homepage des Herstellers. Es werden sowohl Ansprechpartner für die diversen Produkte angeführt, als auch Dokumentation der einzelnen Produkt-Releases angeboten.

Der Kaufvertrag von ObjectIF beinhaltet eine 6monatige Wartung. Diese Wartung umfasst den Hotline Support und die kostenlose Überlassung von Maintenance-Releases und neuen Versionen. Im Rahmen eines optionalen Wartungsvertrages beträgt die Wartungsgebühr für eine Lizenz 18 % des Listenpreises der Lizenzgebühr pro Jahr.

3 UML-Konformität in der Darstellung

Auf Basis der UML-Notation der Version 2.0 werden die Darstellungsmöglichkeiten bewertet. Die Auflistung der Notationsübersicht wurde für diese Arbeit aus dem Buch „UML@Work - Objektorientierte Modellierung mit UML 2“[2] übernommen. Die Konstrukte sind nach den einzelnen Diagrammarten gegliedert.

Symbole zur Bewertung:

+	Darstellung entspricht dem UML 2.0 Standard
-	Darstellung entspricht kaum oder nicht der UML
n	Konstrukt kann nicht dargestellt werden

Klassendiagramm	**Star UML**	**Poseidon for UML**	**Object-eering**	**Visual Paradigm**	**Magic Draw**	**ObjectIF**
Klasse	+	+	+	+	+	+
Klasse mit Abschnitten	+	+	+	+	+	+
Aktive Klasse	+	-	+	+	+	n
Parametrisierte Klasse	+	n	n	+	+	n
Interface Schnittstelle	+	+	+	+	+	+
Angebotenes Interface	+	+	+	+	+	+
Benötigtes Interface	+	+	+	+	+	-
Notiz	+	+	+	+	+	+
Assoziation	+	-	+	+	+	+
Navigierbares Assoziationsende	+	+	+	+	+	+
Nicht navigierbares Assoziations-ende	n	n	n	+	+	n
Qualifizierte Assoziation	+	n	n	+	+	n
Assoziationsklasse	+	+	+	+	+	n
N-äre Assoziation	n	n	+	+	+	n
Schwache Aggregation	+	+	+	+	+	n
Komposition, starke Aggregation	+	+	+	+	+	+
Generalisierung	+	+	+	+	+	-
Abhängigkeit	+	+	+	+	+	+
Abhängigkeit zwischen Mengen von Modellelementen	+	+	+	+	+	+
(Interface-) Realisierungsabhängig-keit	+	+	+	+	+	+
Benutzungsabhängigkeit	+	+	+	+	+	+
Objektdiagramm	**Star UML**	**Poseidon for UML**	**Object-eering**	**Visual Paradigm**	**Magic Draw**	**ObjectIF**
Objekt	+	+	+	+	+	+
Link	+	+	+	+	+	+
Paketdiagramm	**Star UML**	**Poseidon for UML**	**Object-eering**	**Visual Paradigm**	**Magic Draw**	**ObjectIF**
Paket	+	+	+	+	+	+
Paketinhalt	+	+	-	+	+	+
Verschmelzungsbeziehung	+	-	+	+	+	n

[2] Vgl. M. Hitz et al, UML@Work, 3. Auflage, dpunkt.Verlag GmbH, Heidelberg 2005

Import-Beziehung öffentlich	+	-	+	+	+	+
Import-Beziehung privat	+	-	+	+	-	-
Komponentendiagramm	**Star UML**	**Poseidon for UML**	**Object-eering**	**Visual Paradigm**	**Magic Draw**	**ObjectIF**
Komponente	+	+	+	+	+	n
Port, öffentlich	+	+	+	+	+	n
Port, nicht öffentlich	n	n	n	+	+	n
Port mit angebotenem Interface	+	+	+	+	-	n
Port mit benötigtem Interface	+	+	+	+	+	n
Komplexer Port	+	+	+	+	+	n
Komponente mit realisierenden Classifiern	+	-	-	+	+	n
Behavior Port	-	-	n	-	-	n
Verknüpfung über Interfaces	+	+	+	+	+	+
Kompositionsstrukturdiagramm	**Star UML**	**Poseidon for UML**	**Object-eering**	**Visual Paradigm**	**Magic Draw**	**ObjectIF**
Strukturierter Classifier mit interner Struktur	+	+	+	+	+	+
Rolle, existenzabhängig	+	-	+	+	-	-
Rolle, nicht existenzabhängig	n	-	n	+	n	n
Rolle mit anonymer Klasse	+	n	+	+	+	+
Kollaboration	-	+	-	-	+	n
Kollaboration mit Rollen	n	n	n	+	+	n
Kollaboration mit Rollen im Klassendiagramm	n	-	n	-	-	n
Verwendung einer Kollaboration	n	+	-	-	-	n
Konnektor, ungerichtet	n	n	n	+	+	n
Konnektor, gerichtet	n	n	n	n	n	n
Rollenbindung, Parameterbindung	n	+	n	n	n	n
Verteilungsdiagramm	**Star UML**	**Poseidon for UML**	**Object-eering**	**Visual Paradigm**	**Magic Draw**	**ObjectIF**
Artefakt	+	+	+	+	+	n
Knoten	+	+	+	+	+	n
Geräteknoten	+	+	+	+	+	n
Ausführungsumgebungsknoten	+	+	-	+	+	n
Knoten mit zugewiesenem Artefakt	+	+	+	+	+	n
Verteilungsspezifikation	+	+	-	+	+	n
Verteilung	+	+	+	+	+	-
Manifestation	+	+	+	+	+	-
Anwendungsfalldiagramm	**Star UML**	**Poseidon for UML**	**Object-eering**	**Visual Paradigm**	**Magic Draw**	**ObjectIF**
Akteur	+	+	+	+	+	+
Anwendungsfall	+	+	+	+	+	+
Include-Beziehung	+	+	+	+	+	+
Extend-Beziehung	+	+	+	+	+	+
Extend-Beziehung mit Bedingung	+	-	-	-	+	n
Erweiterungsstelle	+	+	+	+	+	n
Anwendungsfall wird durch Classifier realisiert	n	n	n	n	n	n

Classifier besitzt Anwendungsfall	n	n	n	n	n	n
Generalisierung von Akteuren	n	n	+	+	+	+
Generalisierung von Anwendungsfällen	n	n	+	+	+	+
Aktivitätsdiagramm	**Star UML**	**Poseidon for UML**	**Object-eering**	**Visual Paradigm**	**Magic Draw**	**ObjectIF**
Aktivität	n	n	+	+	+	n
Aktion	+	+	+	+	+	+
Initialknoten	+	+	+	+	+	+
Aktivitätsendknoten	+	+	+	+	+	+
Ablaufendknoten	+	+	+	+	+	n
Entscheidungsknoten	+	+	+	+	+	+
Vereinigungsknoten	+	+	+	+	+	+
Parallelisierungsknoten	+	+	+	+	+	+
Synchronsierungsknoten	+	+	+	+	+	+
Objektknoten und Pin	n	-	+	+	+	n
Eingabepin	n	+	+	+	+	n
Ausgabepin	n	+	+	+	+	n
Pufferknoten, persistent	+	+	+	+	+	n
Pufferknoten, transient	+	+	+	+	-	n
Parametersatz	n	n	-	+	+	n
Datenstromparameter	-	-	+	-	-	n
Ausgabeparameter für Ausnahmen	n	n	n	n	n	n
Kontrollflusskante	+	+	+	+	+	+
Objektflusskante	-	+	+	+	+	+
Fortsetzungsmarke	n	n	n	+	-	n
Partition	+	+	+	+	+	+
Expansionsbereich	n	+	n	+	+	n
Ausnahmebehandlungsknoten	n	+	n	+	+	n
Unterbrechungsbereich	n	+	-	+	+	n
Signalsender	+	+	+	+	+	n
Verhaltensaufruf	n	n	n	n	n	n
Ereignisempfänger	+	+	+	+	+	n
Zeitereignisempfänger	n	n	+	+	+	n
Zustandsdiagramm	**Star UML**	**Poseidon for UML**	**Object-eering**	**Visual Paradigm**	**Magic Draw**	**ObjectIF**
Zustand	+	-	+	+	+	+
Zustandsliste	+	-	-	-	-	-
Komplexer Zustand	+	+	+	+	+	+
Entscheidungsknoten	n	-	+	+	+	+
Startzustand	+	+	+	+	+	+
Endzustand	+	+	+	+	+	+
Einstiegspunkt	+	+	n	+	+	+
Ausstiegspunkt	+	+	n	+	+	+
History-Zustand	+	+	+	+	+	+
Tiefer History-Zustand	+	+	+	+	+	n
Verbindungsstelle	+	+	+	+	+	+

Terminierungsknoten	n	+	n	+	+	+
Zustandsautomat	+	-	+	+	+	+
Subautomatenzustand	+	+	+	+	+	+
Gabelung und Vereinigung	+	+	+	+	+	+
Transition	+	+	+	+	+	+
Sequenzdiagramm	**Star UML**	**Poseidon for UML**	**Object-eering**	**Visual Paradigm**	**Magic Draw**	**ObjectIF**
Lebenslinie	+	+	+	+	+	-
Lebenslinie für aktives Objekt	+	+	n	n	n	n
Zerlegung einer Lebenslinie	n	n	+	n	n	n
Referenz auf innere Interaktion einer Lebenslinie	n	n	+	n	+	n
Ausführungsspezifikation direkte/indirekte Ausführung	n	n	+	n	-	n
Interaktionsreferenz	+	+	+	+	-	n
Kombiniertes Fragment	+	+	+	+	-	n
Zustandsvariante/Fortsetzungsmarke	n	n	+	+	n	n
Coregion	n	n	n	n	n	n
Löschsymbol	n	n	+	n	+	+
Zeiteinschränkung / Zeitüberwachung	n	n	n	+	+	n
Zeitdauereinschränkung / Zeitdauerüberwachung	n	n	n	+	+	n
Nachricht synchron	+	+	+	+	+	+
Nachricht asynchron	+	+	+	+	+	+
Antwortnachricht	+	+	+	+	+	+
Verlorene Nachricht	n	+	n	+	n	n
Gefundene Nachricht	n	+	n	+	n	n
Ordnungsbeziehung	n	n	n	n	n	n
Kommunikationsdiagramm	**Star UML**	**Poseidon for UML**	**Object-eering**	**Visual Paradigm**	**Magic Draw**	**ObjectIF**
Lebenslinie	+	-	+	+	+	-
Nachricht	+	-	+	+	+	n
Zeitdiagramm	**Star UML**	**Poseidon for UML**	**Object-eering**	**Visual Paradigm**	**Magic Draw**	**ObjectIF**
Nachricht	n	n	n	+	+	n
Nachrichtenmarke	n	n	n	n	n	n
Zustandsverlauflinie	n	n	n	n	n	n
Werteverlaufslinie	n	n	n	n	n	n
Lebenslinie	n	n	n	+	+	n
Weitere Notationselemente	**Star UML**	**Poseidon for UML**	**Object-eering**	**Visual Paradigm**	**Magic Draw**	**ObjectIF**
Diagrammrahmen	+	+	+	+	+	n
Stereotyp	+	+	n	+	+	n
Metaklasse	+	n	n	+	+	n
Profil	+	n	n	+	-	n
Erweiterung	n	n	n	+	n	-
Profilanwendung	+	n	n	n	n	-

4 Kriterienkatalog

Die Kriterien für die Bewertung der UML-Modellierungswerkzeuge basieren auf der Diplomarbeit „Untersuchung und Vergleich UML-basierter Entwicklungswerkzeuge“ von Oliver Grünsteidl und Jochen Lehner, verfasst an der Universität Wien im Jahr 2000. Einige Kriterien wurden übernommen, überarbeitet, bzw. neu definiert.

4.1 Methodenbasierte Kriterien

a) Modellierung der Objektstruktur-Objektstatik

0. Die Reihenfolge der Komponenten einer Klasse im Klassendiagramm kann der Benutzer bestimmen.
1. Für jedes Attribut kann ein Datentyp festgelegt werden.
2. Für eine Methode kann ein Rückgabetyp angegeben werden, falls eine Methode einen Wert bzw. Objekt zurückliefert.
3. Für eine Methode können Parameter angegeben werden, die beim Methodenaufruf übergeben werden. Dabei kann auch spezifiziert werden, ob es sich um Eingangs-, Ausgangs- oder Durchgangsparameter handelt.
4. Für eine Methode können Vor- und Nachbedingungen angegeben werden, die erfüllt sein müssen (Assertions).

b) Modellierung der Systemstruktur-Systemstatik

1. Auch rekursive Beziehungen können erfasst werden.
2. Einmal erfasste Beziehungen können jederzeit nachträglich verändert werden, sofern die Änderungen mit den bisher erfassten Inhalten verträglich sind.
3. Unabhängig von der Zuordnung von Klassen zu Packages lassen sich Klassen nach bestimmten Gesichtspunkten in Klassenkategorien einteilen.
4. Es werden OCL-Annotationen unterstützt.

c) Dynamisches Objektverhalten-Objektdynamik

1. Für jedes Objekt bzw. jede Klasse (Abstraktion) kann ein Zustandsdiagramm erstellt werden. Dieses ist umkehrbar eindeutig mir der jeweiligen Klasse verbunden.
2. Es können Bedingungen modelliert werden, die einen Zustandsübergang bewirken. Dabei ist ein Bezug auf Attribute eines Objekts möglich.
3. Das Werkzeug kann Aktionen darstellen, die einem Zustandsübergang zugeordnet sind. Dabei ist ein Bezug auf Methoden des Objekts möglich.
4. Das Tool ermöglicht die Verfeinerung von Aktivitäten und Ereignissen im Sinne einer genaueren Beschreibung.
5. Die Darstellung von geschachtelten Zustandsdiagrammen, die eine Generalisierung von Zuständen modellieren, ist möglich. Dabei muss ein Objekt in irgendeinem Zustand des untergeordneten Zustandsdiagramms (Unterstatus) sein.
6. Die Modellierung einer Generalisierung von Ereignissen ist möglich. Diese ist unabhängig von der modellierten Klassenhierarchie.
7. Bei Sequenzdiagrammen kann auf Methoden bzw. Attribute des statischen Objektmodells Bezug genommen werden.

d) Zusammenhang zwischen den Modellen

1. Jedes Zustandsdiagramm ist umkehrbar eindeutig mit einem Objekt bzw. einer Klasse verbunden.
2. Das dynamische Verhalten (Zustandsdiagramm) einer Klasse wird an ihre Unterklasse vererbt.
3. Ereignisse aus der Modellierung des dynamischen Objektverhaltens können Methoden der jeweiligen Klasse zugeordnet werden.
4. Bei der Modellierung von Zuständen im dynamischen Modell kann auf Attribute des Objektmodells Bezug genommen werden.

e) Abstraktionsebenen / Sichten

1. Der Detaillierungsgrad von Klassendiagrammen am Bildschirm lässt sich durch Ausblenden bzw. Einblenden von Attributen und Methoden individuell einstellen.
2. Die Darstellung von Attributen und Methoden ist auf unterschiedlichem Detaillierungsniveau möglich (Ein-/Ausblenden von Typen, Rückgabetypen, Parameter, Parametertypen etc.).
3. Die Angaben über die Sichtbarkeit von Objektkomponenten können für eine Variation der Genauigkeit der Darstellung herangezogen werden (z.B. lediglich Anzeige der öffentlichen Komponenten).
4. Das Detaillierungsniveau der Diagrammanzeige kann sowohl für jedes Diagramm einzeln, als auch generell als Default-Einstellung festgelegt werden.

f) Vorgangsreihenfolge bei der Entwicklung

1. Das Werkzeug zwingt zu einer bestimmten Vorgangsweise, was die Reihenfolge des Anlegens von Modellierungsinformationen betrifft. Klassen etwa, können zunächst nur mit einem Namen angelegt und erst später genauer beschrieben werden.
2. Modellelemente können direkt textuell im Data-Dictionary erfasst und erst später in ein oder mehrere Diagramme eingezeichnet werden.
3. Neu anzulegende Modellelemente können direkt in ein Diagramm eingezeichnet werden, wobei die entscheidenden Informationen automatisch sofort im Data-Dictionary erfasst werden.

g) Unterstützung bei der Methodenanwendung

1. Zur Unterstützung bei der Anwendung von UML lasen sich aus dem Tool Informationen zu deren Funktionen, Notation und Semantik in einer hypertextartigen „Benutzerhilfe“ ausgeben.
2. Die Notationselemente des Tools entsprechen 1:1 jenen der UML, so dass die Verwendung von Basisliteratur zur Methode keine Verwirrung auslöst.
3. Bei der Eingabe von Modellierungsinformationen, die der Konsistenzprüfung bzw. den Integritätsbedingungen nicht standhalten, erscheint eine Warnung und eine Erklärung der entsprechenden Konzepte ist abrufbar.

4.2 Funktionale Kriterien

a) Datenhaltung

1. Modellierungsinformationen, die in mehreren Diagrammen vorkommen, sind in der Basis nicht redundant bzw. nur einmal gespeichert (Integration).
2. Modellierungsinformationen, die in der Datenbasis erfasst sind, müssen nicht zwingend in einem Diagramm vorkommen.
3. Modellinformationen können auch direkt in der Datenbasis gewartet (insbesondere auch gelöscht) werden.
4. Das Datenformat kann als „offen“ bezeichnet werden und ist nicht ausschließlich in diesem Tool anzutreffen.
5. Der Import und Export von Daten über Schnittstellen ist möglich.

b) Browsing-Eigenschaften

1. Das Auffinden von beliebigen Elementen (Klassen, Attribute, Methoden, etc.) durch Eingabe ihres Namens ist möglich.
2. Elemente können auch nach Ähnlichkeitskriterien, durch Pattern Matching, gesucht werden.
3. Die Suche von Elementen kann durch bloße Eingabe der Anfangsbuchstaben vorgenommen werden, wodurch die jeweils zutreffenden Elemente zur Auswahl angeboten werden.
4. Bei der Suche können Einschränkungen vorgenommen werden, etwa durch das Setzten von Filtern.
5. Als Ergebnis einer Suche erhält man das gesuchte Element mit weiteren wichtigen Informationen – z.B. für eine Klasse: Oberklasse, Unterklassen, Beziehungen zu weiteren Klassen, Attribute, Methoden.
6. Die Angaben, die zu einem Modellierungselement gemacht werden, dienen unmittelbar als Link zum Zugriff auf genauere Informationen über diese selbst.
7. Das Werkzeug ermöglicht die Durchführung von Änderungen an beliebigen Modellierungsinformationen direkt aus dem Browser heraus. Diagramme, auf die solche Änderungen Auswirkungen haben, werden sofort automatisch angepasst.
8. Das Werkzeug ermöglicht auch in textueller Form eine übersichtliche Darstellung von Klassenhierarchien.
9. Das Tool ermöglicht auf einfache Art die Navigation innerhalb einer Klassenhierarchie (z.B. durch einfaches Anklicken der Oberklasse).
10. Es gibt die Möglichkeit, für jedes Modellierungselement eine Aufstellung zu erhalten, in welchen übergeordneten Strukturen es vorkommt (z.B. in welchen Klassen kommt das Attribut „GebDatum“ vor?).
11. Alle Modellelemente, die in der Datenbasis gespeichert sind, aber noch in keinem Diagramm vorkommen, können auf einfache weise ausgewählt werden.

c) Navigation im Systemmodell

1. Das Werkzeug erlaubt eine rasche und übersichtliche Navigation innerhalb bzw. zwischen den Darstellungen für einzelne Modellierungsaspekte eines Systems.

2. Es gibt die Möglichkeit, ausgehend von einem Diagramm direkt zur jeweils übergeordneten bzw. verfeinerten Darstellung zu wechseln. Z.B. kann durch einen einfachen Mausklick aus einem Zustandsdiagramm zum entsprechenden Klassendiagramm verzweigt werden.
3. Die einzelnen Elemente eines Diagramms sind nicht nur textuelle bzw. graphische Darstellungen sondern gleichzeitig Hot-Links zum Aufrufen von elementspezifischen, genaueren Informationen.

d) Konsistenzprüfung / Integrität

1. Es können keine Assoziationen, Datenflüsse oder Transaktionen angelegt werden, die ins „Leere“ gehen.
2. Rekursive Vererbungsstrukturen können nicht erstellt werden.
3. Mehrere Modellelemente mit gleichem Namen können in einem Paket nicht angelegt werden.
4. Beim Anlegen von Namen, die bereits bestehenden ähnlich sind, wird der Benutzer gewarnt. Z.B. „GebDatum“ und „Gebdatum“.
5. Das Tool verfügt über eine Funktion, mit der ein bestehendes Modell bzw. der gesamte Datenbestand in einer Datenbasis auf Konsistenz geprüft werden kann. Dabei werden auch Beziehungen zwischen den Teilmodellen bzw. zwischen Diagrammen geprüft.
6. Wird ein Modellierungselement aus einem Diagramm gelöscht, darf es nicht gleich automatisch aus der Datenbasis gelöscht werden (referentielle Integrität).

e) Report-Ausgabe

1. Basierend auf den Daten des Data-Dictionary lassen sich automatisch Auswertungen mit einstellbarem Detaillierungsgrad ausgeben.
2. Für die statische Objektstruktur (Klassen, Attribute, Methoden etc.) können geeignete Reports ausgegeben werden.
3. Für das dynamische Objektverhalten lassen sich geeignete Reports ausgeben,
4. Für die statische Systemstruktur (Vererbungs-, Aggregations-, Assoziationsbeziehungen) lassen sich geeignete Reports erstellen.
5. Auch zur Darstellung des dynamischen Systemverhaltens gibt es Reports (z.B. über message-passing).
6. Ein Report, der über die Ergebnisse einer expliziten Konsistenzprüfung Informationen enthält, lässt sich generieren.
7. Reports können auch einfach auf Grund von ad-hoc Abfragen oder auf andere, frei definierte Weise erstellt werden.
8. Die Ausgabe von Reports in verschiedenen Formaten ist möglich (PDF, PS, usw.).

f) Datenbankschema-Ausgabe

1. Die Generierung eines Datenbankschemas für relationale Datenbanken auf Basis der Informationen des Objektmodells ist möglich.
2. Die Generierung eines objektorientierten Datenbankschemas auf Basis der Informationen des Objektmodells ist möglich.
3. Die Angabe von Schlüsselattributen für eine spätere DB-Schema-Ausgabe ist möglich.

4. Zusätzlich zu den Datentypen für die bei der Codegenerierung zu verwendende Zielsprache können bei den Attributen auch Domains angegeben werden.
5. Alle datenbankrelevanten Informationen aus der Klassenstruktur und der statischen Struktur des Anwendungssystems finden sich in den generierten Datenbankschemata wieder (Attribute, Beziehungen, etc.).

g) Diagramm-Ausgabe

1. Der Diagrammausdruck kann als präsentationsfähig bezeichnet werden.
2. Alle Diagramme lassen sich so ausdrucken, dass sie durch einfaches Zusammenleben von A4 Blättern übersichtliche Dokumentationen ergeben.
3. Die gedruckten Diagramme entsprechen genau der Darstellung am Bildschirm.
4. Das Abstraktionsniveau lässt sich für die Diagrammausgabe frei wählen (z.B. nur Klassennamen anzeigen / Klassen, Attribute u. Methoden anzeigen etc.).
5. Bei der Diagrammausgabe sind die Ausrichtung (Quer- oder Hochformat) und Skalierung der Darstellung einstellbar.

h) Offenheit-Import / Export

1. Das Tool ist ein Modul aus der Produktpalette des Herstellers, welches mit Zusatzprodukten des Herstellers zusammenarbeitet und auf diese Weise modulartig erweitert werden kann.
2. Das Tool schöpft die von der Plattform gebotenen Möglichkeiten des Datenaustauschs sowohl für Text als auch für graphische Darstellungen aus (z.B. Cut&Paste von Grafiken und Texten über die Zwischenablage).
3. In den Graphikeditor können auch geläufige toolfremde Grafikformate eingelesen werden.
4. In den Zeicheneditoren können auch geläufige Textformate direkt eingelesen werden.
5. Es gibt bereits Zusatztools von Drittherstellern, die Ausgaben des Tools einlesen und weiterverarbeiten können.

i) Codegenerierung

1. Die automatische Codegenerierung für eine oder mehrere in Frage kommende objektorientierte Zielsprache ist möglich.
2. Für jede im Modell vorhandene Klasse wird im Sourcecode der Zielsprache eine entsprechende Codierung vorgenommen.
3. Attribute werden inklusive ihrer Typen in die Zielsprache umgesetzt.
4. Methoden werden inklusive ihrer Argumente (inkl. Typen) und Rückgabetypen in den entsprechenden Sourcecode richtig umgesetzt.
5. Beziehungen zwischen Klassen werden im resultierenden Sourcecode richtig umgesetzt.
6. Für Methoden werden Sourcecode-Gerüste produziert, die noch durch den Inhalt ergänzt werden müssen.
7. Zugriffsbeschränkungen werden auch im Programmcode berücksichtigt.

8. Strukturierungen oberhalb des Niveaus der Klassen finden sich in der physischen Struktur des Codes wieder (z.B. mehrere Module führen zu einzelnen Source-Dateien).
9. Das Ausmaß der Codegenerierung lässt sich durch Parameter bestimmen (z.B. Defaultwerte, Datentypen, Rückgabetypen, Argumente, Zugriffsbeschränkungen, Präfix für Klassennamen, etc.).
10. Wenn generierter Code händisch erweitert wurde und die automatische Codegenerierung erneut angestoßen wird, bleibt der eigene (user added) Code erhalten (z.B. Kennzeichnung durch toolseitig vergebenen Kommentaren).
11. Inhalte von Methoden lassen sich schon in der textuellen Beschreibung der Methode eintragen und werden in den Code übernommen.

j) Reverse-Engineering

1. Bestehender Programmcode kann mit dem Tool eingelesen werden. Dieses erzeugt aus den Code-Informationen die entsprechenden A/D-Diagramme.
2. Es kann auch Code eingelesen werden, der nicht mit dem Tool selbst generiert wurde.
3. Codefragmente, die über Klassendefinitionen, Komponenten und Beziehungen hinausgehen (z.B. der Inhalt von Methoden) können in einem Editor bearbeitet und erneut als Code gespeichert werden.

k) Test / Verifikation

1. Das Tool erlaubt die textuelle Beschreibung von Testfällen für einzelne Objekte bzw. Methoden
2. Das Tool ermöglichte eine Simulation auf Modellebene, bei der das Senden von Nachrichten simuliert und das entsprechende Objektverhalten dargestellt wird.

l) Maßzahlen und Statistiken

1. Das Tool kann auf Basis der Daten des Repository geeignete Software-Maßzahlen berechnen, etwa die Tiefe von Vererbungshierarchien oder Kopplungsmaßzahlen etc.
2. Die automatische Ausgabe von Statistiken (Anzahl modellierter Klassen, Attribute, Methoden, etc.) ist möglich.
3. Die Ausgabe von Informationen über den Projektstatus (Mitarbeiter, Zugriffsberechtigungen, letzte Änderungen etc.) ist möglich.

m) Multi-User-Eigenschaften

1. Das Tool kann von mehreren Benutzern gleichzeitig verwendet werden.
2. Die Benutzer des Werkzeugs können – unabhängig vom Basis-Betriebssystem – in User und Gruppen eingeteilt werden (tooleigenes „Userkonzept“).
3. Es gibt einen Konfliktlösungsmechanismus, der verhindert, dass kein Benutzer die Arbeit eines anderen versehentlich zerstört (Locking).
4. Der Umfang des Lockings ist einstellbar – z.B. von Modellebene bis zur Ebene eines Attributs.

n) Security

1. Jeder Benutzer muss sich mit seinem User-Namen und einem Passwort ausweisen, bevor er das Werkzeug nutzen kann (Login/Logout).
2. Zugriffsrechte für Benutzer können auf User- oder Gruppenebene festgelegt werden.
3. Über ein automatisch mitgeschriebenes Protokoll können Änderungen auch im Nachhinein auf dem dafür verantwortlichen Benutzer zurückgeführt werden (Logging).
4. Das Tool verfügt über eine geeignete Möglichkeit zur Datensicherung, die auch automatisierbar ist.
5. Für den Fall von Systemabstürzen verfügt das Werkzeug über einen geeigneten Recovery-Mechanismus, der die Integrität des Programms und der Daten gewährleistet.
6. Jeder Benutzer kann Elemente eines Modells vorübergehend vor Änderungen schützen (einfrieren), auch falls er nicht gerade daran arbeitet und eingeloggt ist.
7. Im Falle von Bedienungsfehlern verbleibt das Werkzeug in einem definierten Zustand und gibt eine Fehlermeldung aus.
8. Unbeabsichtigte Löschungen können durch „Bestätigungsabfragen“ verhindert werden.
9. Löschungen können bis einem Gewissen Grad automatisch rückgängig gemacht werden (generelle Undo-Funktion).

o) Multi-Projekt-Eigenschaften / Versionsmanagement

1. Das Werkzeug ermöglicht die zeitlich parallele Abwicklung von mehreren Projekten.
2. Der Austausch von Modellierungsinformationen zwischen einzelnen Projektbasen ist möglich.
3. Der Bezug auf bereits in anderen Projekten gespeicherten Modellierungsinformationen (Referenzen) ist möglich. Somit können bereits erstellte Modelle wieder verwendet werden, ohne diese kopieren zu müssen.
4. Das Werkzeug erlaubt die gleichzeitige Verwaltung von mehreren verschiedenen Versionen eines Systemmodells.
5. Das Tool kann automatisch Änderungen einer Version gegenüber der Ausgangsversion anzeigen.

4.3 Ergonomie

a) Benutzeroberfläche

1. Das allgemeine Erscheinungsbild ist optisch ansprechend und übersichtlich.
2. Das GUI ist so implementiert, dass das Look-and-Feel der entsprechenden Benutzeroberfläche vorhanden ist.
3. Dialoge und Eingabemasken sind übersichtlich und zweckmäßig zu bedienen.
4. Die Namensgebung von Menüaktionen und Auswahlmöglichkeiten sind weitgehend selbstbeschreibend, einheitlich und widerspruchsfrei.
5. Momentan nicht verfügbare Menübefehle oder Optionen sind besonders dargestellt.
6. Im Falle von länger dauernden Aktionen erhält der Benutzer Informationen über Status und Verarbeitungsfortschritt (z.B. Fortschrittsbalken, Sanduhr).
7. Die Anzeige von Listen am Bildschirm ist einheitlich und übersichtlich.

b) Tool-Dokumentation

1. Es gibt ein ausführliches, gedrucktes Handbuch zur Bedienung des Tools.
2. Es gibt schriftliche Anleitungen zur Installation.
3. Das Handbuch ist in der jeweiligen Landessprache (Deutsch) verfasst.
4. Es gibt ein Kurz-Handbuch für versierte Benutzer.
5. Es gibt ein „Erste-Schritte“ Handbuch für unerfahrene Benutzer.
6. Das Handbuch hat ein Inhaltsverzeichnis und einen Stichwortindex bzw. Referenzteil.
7. Das Handbuch lässt sich leicht ergänzen, Teile lassen sich leicht austauschen.
8. Es gibt ein hypertextartiges Onlinehandbuch.
9. Das Onlinehandbuch ist direkt aus dem Tool heraus abrufbar.
10. Das Onlinehandbuch kann im Hintergrund offen gehalten werden.
11. Mit dem Tool wurden fertige Beispiele geliefert, die als Anregung und zur Einführung verwendet werden können, und die sich zum Selbststudium eignen.
12. Wichtige Informationen, die sich erst nach Vorliegen der Dokumentation ergeben, sind in Textdateien vorhanden, auf die auch die Dokumentation verweist.

c) Usability

1. Der Abbruch von Aktionen ist zu jeder Zeit möglich.
2. Im Fehlerfall wird der Benutzer durch eine informative und weiterführende Fehlermeldung benachrichtigt. Er erhält auf Wunsch auch direkt Hilfe zur aktuellen Situation.
3. Für wichtige Menüfunktionen gibt es Abkürzungs-Tastaturkombinationen (Tastatur-Shortcuts).
4. Funktionstasten können mit oft benötigten Befehlsfolgen belegt werden.
5. Das Tool bietet für ein rasches Arbeiten Kontextmenüs mit den jeweils anwendbaren, wichtigsten Befehlen an.

6. Die Eingabe von ungültigen Werten bzw. Ausführung ungültiger Operationen führt zu einer kontrollierten Fehlerbehandlung: Es erscheint eine Fehlermeldung und der kontrollierte Programmablauf wird fortgesetzt.
7. Es gibt Symbolleisten (Toolbar, Buttonbar), aus denen man häufig benötigte Befehle sofort abrufen kann – etwa eine Werkzeugleiste mit Symbolen zur Objektmodellierung. Die Symbolleisten sind vom Benutzer individuell konfigurierbar.
8. In einer Statuszeile finden sich hilfsreiche Informationen – etwa über den gerade auszuführenden Befehl und die weitere Vorgangsweise.
9. Mehrfenstertechnik wird unterstützt d.h. mehrere Darstellungen desselben Projekts, z.B. verschiedene Diagramme können gleichzeitig nebeneinander betrachtet werden.
10. Die Selektion aus Listen ist auf einfache Weise möglich (Einfach- und Mehrfachauswahl).
11. Es gibt eine Version des Werkzeugs in deutscher und englischer Sprache.

d) Hilfefunktionen

1. Das Tool bietet eine kontextsensitive Hilfefunktion, die leicht erreichbar ist (z.B. F1).
2. Das Werkzeug hat eine Hypertexthilfe, welche alle Hilfethemen umfasst.
3. Die Toolhilfe ermöglicht das Suchen nach bestimmten Hilfethemen.
4. Das Werkzeug bietet eine Hot-Help an: Wird die Maus auf ein Element, bzw. einen Menübefehl bewegt, erscheint dazu auf Wunsch die Hilfe.

e) Graphikeditor / Darstellung

1. Die Symbole sind in einem bestimmten Umfang standardmäßig anpassbar (als Voreinstellung).
2. Zusätzlich zu den Diagrammen aus UML können „ freie“ Diagramme aus den gängigsten Umrissformen (Rechtecke, Ovale, Linien, etc.) mit Beschriftungen erstellt werden.
3. Die Darstellung der Diagramme am Bildschirm entspricht dem späteren Ausdruck (WYSIWIG).
4. Für Notationselemente können verschiedene, selbst definierbare Farben voreingestellt werden.
5. Es gibt eine Funktion zur horizontalen bzw. vertikalen Ausrichtung von Text oder Symbolen (Alignment).
6. Die Bildschirmdarstellung ganzer Diagramme oder Teile eines Diagramms kann (stufenlos) vergrößert und verkleinert werden (Zoom).
7. Es gibt eine Funktion zum Rotieren von Symbolen und vor allem von Texten.
8. Es existiert eine Funktion, die ein Diagramm automatisch in der optimalen Größe und ohne Überlappungen von Elementen am Schirm anzeigt.
9. Da Layout eines Diagramms, welches man speichert und wieder lädt, wird beibehalten.
10. Die Schriftgröße zur Beschreibung von Diagrammelementen ist frei wählbar und die Schrift selbst gut lesbar.

11. Die Größe aller Notationselemente ist einstellbar, aber insgesamt einheitlich und nicht etwa von der Länge der darin enthaltenen Beschreibung (z.B. Klassenname) abhängig.
12. Ein Zeilenumbruch innerhalb einer Elementbeschriftung ist möglich.
13. Für alle Linien (bei Assoziationen) können auch nachträglich beliebige Eckpunkte festgelegt werden, um Überschneidungen zu vermeiden.
14. Der Editor erlaubt die Einstellung von Strichstärke und Linienart.
15. Es gibt eine Funktion zum „Neuzeichnen“ des Bildschirminhaltes durch das System (Refresh).

f) Graphikeditor / Bedienung

1. Das Einzeichnen von Notationselementen in ein Diagramm erfolgt komfortabel durch einfaches Aufziehen mit der Maus oder durch Auswahl aus einem Kontextmenü.
2. Sofort nach dem Anlegen eines Notationselements kann dieses textuell beschrieben werden (z.B. Klassen-, Attribut- und Methodenname).
3. Eine Funktion zum vielfachen Einfügen desselben Elements hintereinander steht zur Verfügung (Tool halten).
4. Notationselemente können durch einfaches drag-and-drop an eine beliebige Stelle eines Diagramms verschoben werden.
5. Der Zeichenbereich für Diagramme ist „unendlich“ groß. Werden Diagramme größer als eine Bildschirmseite, kann horizontal und vertikal gescrollt werden.
6. Zum Zweck einer automatischen, horizontalen, bzw. vertikalen Ausrichtung von Diagrammelementen kann ein frei definierbares Raster angelegt werden (Grid Alignment).
7. Auf Wunsch erscheinen Linien und Symbole exakt dort, wo man sie haben will.
8. Man kann mehrere Symbole gruppieren (z.B. durch Umrahmen mit dem Maus-Cursor), um darauf eine Operation auszuführen (z.B. Verschieben).
9. Das Abbrechen einer Zeichenaktion (z.B. irrtümliches Aufziehen einer Assoziation zu einer noch nicht gezeichneten, zweiten Klasse) ist einfach durchführbar.
10. Es gibt eine Undo-Funktion, die den letzten Bearbeitungsschritt (oder mehrere) rückgängig macht.
11. Es gibt eine Redo-Funktion, die den letzten Bearbeitungsschritt wiederholt.
12. Die allgemeine Bedienung des Graphikeditors verdient das Prädikat „intuitiv“.
13. Im Graphikeditor wird das Kopieren/Ausschneiden und Einfügen von Symbolen unterstütz (Copy/Cut und Paste).
14. Symbole lassen sich leicht wieder aus einem Diagramm entfernen, etwa durch Markieren und Betätigen der „Enft“-Taste.

g) Performance

1. Bei der Darstellung der Diagramme und deren Manipulation (Zoomen, Scrollen, etc.) erfolgt der Bildaufbau in Anbetracht der zur Verfügung stehenden Hardwareumgebung ausreichen schnell.
2. Die Zeit, die das Tool zum Laden und Speichern von Projekten, Diagrammen oder Spezifikationen benötigt, erlaubt ein zügiges Arbeiten.

3. Suchergebnisse stehen, in Anbetracht des bereits erfassten Datenvolumens, rasch zur Verfügung.
4. Die automatische Codegenerierung erfolgt ausreichend schnell.

h) Adaptivität

1. Das Werkzeug kann dem Lernfortschritt der Anwender angepasst werden, z.B. zur Verfügungstellung unterschiedlich detaillierter Menüs (Ausblenden von komplexen Befehlen), Ein- und Ausschalten von Sicherheitsabfragen und Hinweisdialogen, Expert oder Apprentice-Modus etc.
2. Die Oberfläche lässt sich individuell gestalten (Symbolleisten, Funktionstastenbelegung, etc.) und kann für jeden Benutzer spezifisch eingestellt werden.
3. Die Struktur der Menüs kann vom Anwender verändert werden (neue Befehle einfügen, Befehle umbenennen, etc.).
4. Es gibt eine Möglichkeit, benutzerdefinierbare Befehlsfolge aufzuzeichnen und später abzurufen (Makros).
5. Zur Erstellung von Dokumenten gibt es eine Scriptsprache, mit der Ausgaben frei gestaltbar sind. Alle Modellelemente können dabei als Variablen verwendet werden.
6. Das Werkzeug wird im Sourcecode ausgeliefert, was ein Maximum an Anpassungsmöglichkeiten bietet.

5 Evaluationen

Die Ergebnisse der Auswertung beziehen sich auf den Kriterienkatalog aus dem vorherigen Kaptitel. Jeder Aspekt wird einerseits allgemein beschrieben, zusätzlich findet man eine tabellarische Bewertung der einzelnen Unterpunkte aus dem Katalog, gemäß folgendem Schema:

+	Funktion ist gut umgesetzt, bzw. vorhanden
o	Funktion ist mittelmäßig umgesetzt, bzw. teilweise nicht vorhanden
-	Funktion ist schlecht umgesetzt, bzw. nicht vorhanden

5.1 StarUML

5.1.1. Methodenbasierte Kriterien

a) Modellierung der Objektstruktur - Objektstatik

Das Programm speichert Informationen in einem Repository, das im Browserfenster *Model Explorer* einsehbar ist. In diesem *Model Explorer*, der in Form einer Baumstruktur aufgebaut ist, können alle Informationen verwaltet werden. Hieraus folgt, dass Elemente im Repository existieren können, die in keinem Diagramm vorkommen. StarUML achtet beim Hinzufügen neuer Objekte auf eventuelle Namenskonflikte. Die Attribute haben einen bestimmten Typ, einen Namen und einen Initialwert. Außerdem sind die verschiedenen Arten der Sichtbarkeit (private, public, protected, package) schon vordefiniert. Methoden besitzen Parameter, Sichtbarkeit, Namen und Rückgabetypen.

1	2	3	4	5
+	+	+	+	+

b) Modellierung der Systemstruktur - Systemstatik

Mit StarUML können auch rekursive Beziehungen erfasst werden. Die Einteilung der Klassen erfolgt mittels Packages. Constraints werden unterstützt und können im Constraint-Editor definiert werden.

1	2	3	4
+	+	o	o

c) Dynamisches Objektverhalten – Objektdynamik

Durch das Markieren der gewünschten Klasse und einen Rechtsklick können Klassen eindeutig mit Zustandsdiagrammen verbunden werden. Außerdem erlaubt StarUML *Entry-Actions, Do-Actions* und *Exit-Actions* zu definieren. Ebenso können Aktionen angegeben werden, die Zustandsübergängen zugeordnet sind. Zustände können geschachtelt werden. Zustände können sehr komfortabel ver-

feinert werden. Bei Sequenzdiagrammen kann auf Methoden bzw. Attribute des statischen Objektmodells Bezug genommen werden.

1	2	3	4	5	6	7
+	+	+	+	+	-	+

d) Zusammenhang zwischen den Modellen

Nach dem Anlegen eines Zustandsdiagramms für eine bestimmte Klasse, ist dieser der Klasse eindeutig zugeordnet. Auch im *Model Explorer* werden Zustandsdiagramme unter den jeweiligen Klassen angezeigt und nicht zusammengefasst dargestellt. Besitzt eine Klasse Unterklassen, so ist jedem dieser Klassen ein Zustandsdiagramm zugeordnet.

1	2	3	4
+	+	+	-

e) Abstraktionsebenen / Sichten

Mit StarUML ist es möglich, die einzelnen Elemente in verschiedenen Sichten darstellen zu lassen. Es können Attribute und Methoden ein- bzw. ausgeblendet werden. Ebenso kann die Anzeigenart der Stereotypen verändert werden. Die Sichtbarkeit kann ebenso wie die Eigenschaften einer Klasse bzw. Attributen ausgeblendet werden.

1	2	3	4
+	+	+	-

f) Vorgangsreihenfolge bei der Entwicklung

StarUML zwingt den Benutzer nicht, eine bestimmte Reihenfolge beim Anlegen eines Modells einzuhalten. Es existiert somit kein Vorgehensmodell. Bei einer Klasse können Detailinformationen in einem späteren Stadium im Projekt angegeben werden. Der Benutzer kann die Informationen auch direkt im Data-Dictionary erfassen und später in ein oder mehrere Diagramme einzeichnen. Ebenso werden Informationen, automatisch im Data-Dictionary erfasst.

1	2	3
+	+	+

g) Unterstützung bei der Methodenanwendung

UML-spezifische Informationen werden mit StarUML nicht mitgeliefert. Auch auf der Homepage des Herstellers findet man nur wenig Literatur zu UML und deren Funktionen, Notationen oder Semantik. Einige wenige Notationselemente unterscheiden sich vom UML-Standard, Verwirrung kommt aber bei der Verwendung von Basisliteratur dadurch nicht auf.

Eine Konsistenz- und Integritätsprüfung sind vorhanden, es werden aber nur die Anzahl der Fehler ausgegeben, jedoch nicht weiterführende Erklärungen. Es werden keine weiteren Informationen über die Fehlerquellen ausgegeben.

1	2	3
o	o	o

5.1.2. Funktionale Kriterien

a) Datenhaltung

Alle Modellierungsinformationen werden in einem Repository gespeichert. Die Daten werden in einem Baummenü übersichtlich dargestellt und können im *Model Explorer* auch manipuliert werden. Daten, die in mehreren Diagrammen vorkommen, werden im Datenstamm nicht redundant gespeichert. Modellierungsinformationen können auch direkt im *Model Explorer* hinzugefügt und bearbeitet werden. Diese müssen aber nicht zwingend in einem Diagramm vorkommen. Werden Modellierungsinformationen in Diagrammen gelöscht, bleiben diese aber noch im Repository erhalten.

Die Daten werden als UML-File abgespeichert. Inwieweit andere UML-Werkzeuge diese Daten lesen bzw. manipulieren können, ist nicht bekannt. Es ist aber möglich die Daten als XMI-File zu exportieren, auf die dann auch die meisten UML-Werkzeuge zugreifen können. Der Import von Daten ist möglich. Hierzu können auch Daten des UML-Werkzeugs Rational Rose importiert werden. Die Daten müssen im XMI-Format vorliegen, um importiert werden zu können.

1	2	3	4	5
+	+	+	+	+

b) Browsing-Eigenschaften

Das Programm besitzt eine Suchfunktion, mit der sich beliebige Elemente eines Modells, egal welcher Art, leicht auffinden lassen. Durch die Eingabe eines Suchbegriffs, enthält der Benutzer eine Liste mit den gefundenen Elementen und eine Beschreibung, in welchem Diagramm der gefundene Begriff vorkommt. Mit einem Doppelklick auf ein Suchergebnis springt das Programm zum entsprechendem Element im Repository und es wird auch das dazugehörige Diagramm angezeigt. Eine Suche nach Ähnlichkeitskriterien ist nicht möglich.

Es können auch Einschränkungen bei der Suche vorgenommen werden. Durch das Setzen von Filtern ist es möglich nach bestimmten Elementtypen wie z.B. Paketen, Knoten oder Klassen zu suchen. In der Ergebnisliste werden auch weitere Informationen zu den Suchtreffern angezeigt. Es wird angezeigt, um welchen Elementtyp es sich handelt und in welchem Diagramm er vorkommt. Genauere Informationen werden dem Benutzer nicht angezeigt.

Änderungen, die im *Model Explorer* gemacht wurden (z.B. Änderungen der Bezeichnungen), werden automatisch in den Diagrammen übernommen. Textuelle Darstellungen von Klassenhierarchien können nicht erstellt werden.

1	2	3	4	5	6	7	8	9	10	11
+	-	-	+	o	-	+	-	-	o	+

c) Navigation im Systemmodell

Die Navigation zwischen den einzelnen Modellen erfolg über den *Diagram Explorer*. Hier werden zu jedem Diagrammtyp alle vorhandenen Diagramme angeführt. Die Navigation erfolgt über eine Baumstruktur. Zustandsdiagramme sind im *Model Explorer* Klassen untergeordnet und können nur über das Baummenü erreicht werden. Beim Markieren von Elementen durch die Maus werden weiterführende Informationen in der Statusleiste angezeigt.

1	2	3
+	o	+

d) Konsistenzprüfung / Integrität

Will der Benutzer eine Assoziation anlegen, die ins Leere geht, liefert StarUML eine Fehlermeldung. Rekursive Vererbungen können modelliert werden, bei der automatischen Überprüfung des Modells gibt es aber eine Fehlermeldung. Will man Klassen mit gleicher Bezeichnung anlegen, führt dies zu einer Fehlermeldung und die Aktion wird abgebrochen. Auf Namensähnlichkeiten wird nicht geprüft. Werden Modellelemente aus einem Diagramm gelöscht, bleiben diese noch im Repository erhalten.

1	2	3	4	5	6
+	o	+	-	o	+

e) Report-Ausgabe

Reports können mit dem StarUML Generator erstellt werden. Bei der aktuell getesteten Version war kein solcher Generator aufzufinden. Wie detailliert diese Reports erstellt sind, konnte nicht eruiert werden. Reports können in verschiedenen Formaten erstellt werden. Die Report Ausgabe konnte nicht detailliert getestet werden. Deshalb sind einige Kriterien mit „?“ gekennzeichnet.

1	2	3	4	5	6	7	8
o	?	?	?	?	?	?	+

f) Datenbankschema-Ausgabe

Mit StarUML ist es nicht möglich, ein Datenbankschema für relationale bzw. objektorientierte Datenbanken zu generieren.

1	2	3	4	5
-	-	-	-	-

g) Diagramm-Ausgabe

Diagramme können automatisch an die gewählte Papiergröße angepasst werden. Die gedruckten Diagramme entsprechen genau der Darstellung am Bildschirm. Es ist nicht möglich, das Abstraktionsniveau einzustellen. Die gewünschte Skalierung der Darstellung kann vor dem Drucken ausgewählt werden.

1	2	3	4	5
+	o	+	-	+

h) Offenheit – Import / Export

Der Hersteller bietet vier weitere Zusatzprodukte an, die in StarUML integriert werden können. Hierbei handelt es sich um Module für Reports und ER-Diagramme. Funktionen der Zwischenablage werden unterstützt. Es ist nicht möglich, Text- bzw. Grafikformate in den Zeicheneditor zu kopieren. StarUML kann Daten in einem auch für andere Anwendungen lesbaren Format ausgeben (XMI, JPEG, WMF). So ist eine Kompatibilität mit anderen Produkten garantiert.

1	2	3	4	5
+	o	+	-	+

i) Codegenerierung

Die Codegenerierung ist für die Programmiersprachen Java, C#, C++ möglich. Für jede im Modell vorhandene Klasse wird im Sourcecode eine entsprechende Codierung vorgenommen. Die angegebenen Methoden und Typen werden mit der Codegenerierung übernommen. Für Methoden werden Blöcke erstellt, die nur noch ergänzt werden müssen. Sichtbarkeit von Klassen wird richtig umgesetzt. Für jede im Klassenmodell erstellte Klasse wird eine eigene Source-Datei erstellt. Das Ausmaß der Codegenerierung lässt sich nur durch Angabe des Modells und der bestehenden Klassen steuern. Bei einer erneuten Codegenerierung werden die vorherigen Dateien automatisch überschrieben.

1	2	3	4	5	6	7	8	9	10	11
+	+	+	+	-	+	+	+	o	-	-

j) Reverse-Engineering

Ein Reverse Engineering mit Java, C++ und C# Quelldateien ist möglich. Entsprechende Diagramme werden automatisch erzeugt. Auch ein toolfremder Code kann eingelesen werden. Die entsprechenden Klassen werden erzeugt. Auch die Typen, Parameter u.ä. werden übernommen. Beim Einlesen von Code innerhalb einer Methode traten unbekannte Fehler auf. Der Vorgang wurde abgebrochen.

1	2	3
+	+	-

k) Test / Verifikation

Eine Möglichkeit zum Testen bzw. zur Simulation auf Modellebene wird von StarUML nicht unterstützt.

1	2
-	-

l) Maßzahlen und Statistiken

Das Berechen von Maßzahlen und Statistiken auf Basis des Repositorys wird nicht zur Verfügung gestellt. Es wäre möglich, extern Berechnungen durchzuführen, da über ein kostenloses Add-In auch Excel-Dateien erzeugt werden können.

1	2	3
-	-	-

m) Multi-User – Eigenschaften

Eine Userverwaltung ist in StarUML nicht vorhanden. Konfliktlösungsmechanismen werden nur vom darunter liegenden Betriebssystem umgesetzt.

1	2	3	3
o	-	-	-

n) Security

Wie schon erwähnt ist eine Multi-User Eigenschaft nicht implementiert. Deshalb braucht sich der Benutzer nach dem Programmstart nicht anmelden um mit diesem Tool arbeiten zu können. Auch ein Logging der durchgeführten Änderungen wird nicht durchgeführt. Besondere Mechanismen zur Datensicherung oder Recovery Mechanismen sind nicht enthalten.

Bedienungsfehler werden vom System abgefangen und entsprechende Fehlermeldungen werden abgesetzt. Informationen können aus dem Repository nur mit dem Drücken einer speziellen Tastaturfolge (Strg + Entf) gelöscht werden. Über die Undo-Funktion können Löschungen wieder rückgängig gemacht werden.

1	2	3	4	5	6	7	8	9
-	-	-	-	-	-	+	o	+

o) Multi-Projekt – Eigenschaften / Versionsmanagement

Zeitgleich kann immer nur ein Projekt geöffnet sein. Außer man hat zwei Instanzen des Programms laufen. Der Austausch von Informationen ist über die Import/Export-Funktion möglich. Jedoch können nur ganze Modelle Ausgetauscht werden und keine Teilmodelle. Referenzen über Modellierungsinformationen aus anderen Projekten können nicht angesprochen werden.

Versionen können nur bei der Ausgabe von Reports angegeben werden. StarUML besitzt kein integriertes Versionsmanagement.

1	2	3	4	5
o	o	-	-	-

5.1.3. Ergonomie

a) Benutzeroberfläche

StarUML besitzt eine für Windows typische Benutzeroberfläche, die vergleichbar mit anderen Entwicklungsumgebungen ist. Alle Bildschirminhalte sind übersichtlich angeordnet und können vom Benutzer individuell eingerichtet werden. Der Benutzer kann durch das Mauswerkzeug die Größe und die Position der einzelnen Teilbereiche des Hauptfensters (Toolbox-, Eigenschafts-, Diagramm- und Ausgabefenster) an seine Bedürfnisse anpassen.

Die Systemdialoge sind für den Benutzer verständlich und klar formuliert. Die Eingabemasken sind einfach gehalten und sind nicht überladen. Der Aufbau des Menüs entspricht ebenso den Windows-Standard (File, Edit, Format, ..., Help). Momentan nicht verfügbare Funktionen werden mit geringer Sichtbarkeit, grau dargestellt und können vom Benutzer nicht ausgewählt werden. Um den Benutzer über eine längere, vom Computer durchgeführte Aktion zu informieren, verwandelt sich der Mauszeiger in eine Sanduhr und Fortschrittsanzeigen werden eingeblendet.

1	2	3	4	5	6	7
+	+	+	+	+	+	+

b) Tool-Dokumentation

StarUML bietet eine ausführliche Hilfefunktion. Die Hilfe ist aus dem Programm abrufbar und wird schon bei der Programminstallation mit installiert. Ein druckbares Handbuch kann kostenlos von der Herstellerseite geladen werden. Das Hilfesystem besitzt ein Inhaltsverzeichnis und ist nach Kapiteln zusammengestellt. Es ist möglich auch nach Stichwörtern zu suchen. Die Hilfe ist aber nur in englischer Sprache verfügbar. Ebenso verfügbar sind Hilfestellungen zu diversen Add-Ins, wie z.B. Informationen für diverse Programmiersprachen wie C#, C++ oder Java. Ebenso vorhanden sind vier Beispielprojekte, die sich für die Einarbeitung in StarUML gut eignen. Vermisst wird aber eine schriftliche Anleitung zur Installation, sowie ein „Erste-Schritte“ Handbuch für unerfahrene Benutzer.

1	2	3	4	5	6	7	8	9	10	11	12
+	-	-	-	-	+	-	+	+	+	+	+

c) Usability

Einige Systemoperationen, wie das Laden eines Projektes oder die Prüfung des Modells auf deren Richtigkeit, kann der Benutzer nicht unterbrechen. Der Benutzer bekommt im Fehlerfall zwar informative Fehlermeldungen, ein direkter Ver-

weis zur aktuellen Hilfestellung wird aber nicht angeboten. Wie aus anderen Anwendungen bekannt, hat der Benutzer die Möglichkeit auch bei StarUML mit Tastatur-Shortcuts zu arbeiten. Für die am meisten gebrauchten Programmfunktonen sind Tastatur-Shortcuts vorhanden. Funktionstasten finden zwar als Shortcuts Verwendung, können jedoch nicht extra belegt werden.

Das Programm stellt in jeder Situation ein entsprechend angepasstes Kontextmenü bereit, das der Benutzer über die rechte Maustaste aufrufen kann. Ungültige Eingaben werden von StarUML abgefangen und Fehlermeldungen werden ausgegeben. StarUML verfügt über mehrere Symbolleisten, die vom Benutzer individuell angepasst werden können. Die Icons ähneln den Symbolen anderer Produkte, so sind die meisten Funktionen selbstbeschreibend. Ebenso kann die Größe und die Position der Symbolleisten verändert werden. Eine Statuszeile zeigt den Fortschritt bei längeren Prozessen an. Die verschieden Diagramme eines Projektes können zwar gleichzeitig geöffnet werden, es ist jedoch nicht möglich diese nebeneinander zu betrachten.

Eine Mehrfachauswahl ist mit gedrückter Shift-Taste möglich. StarUML ist nicht in deutscher Sprache verfügbar.

1	2	3	4	5	6	7	8	9	10	11
o	o	+	-	+	+	+	+	o	+	-

d) Hilfefunktionen

Die Hilfe kann über die Funktionstaste F1 aufgerufen werden. Man gelangt zum Windows-Helpfile, mit dem man nach Stichwörtern suchen kann. Ebenso vorhanden ist die Hot-Help Funktion.

1	2	3	4
+	+	+	+

e) Graphikeditor / Darstellung

Das Aussehen der Symbole ist vorgegeben und kann vom Benutzer nicht verändert werden. Alle Notationselemente sind UML-konform dargestellt. Es ist möglich eigene Umrissformen (Rechtecke, Ellipsen) zu verwenden, die in der Toolbox vorgegeben sind. Kommentare bzw. textuelle Anmerkungen können individuell in Diagrammen hinzugefügt werden. Die Darstellung der Diagramme am Bildschirm entspricht genau dem späteren Ausdruck. Vor dem Drucken bekommt der Benutzer eine Voransicht der Diagramme. Es ist möglich, die einzelnen Elemente in den Diagrammen bezüglich deren Farbe und Schriftart zu verändern. Die Schriftgröße und der Hintergrund kann auch angepasst werden. Elemente können automatisch horizontal bzw. vertikal ausgerichtet werden. Das Vergrößern bzw. Verkleinern der Bildschirmdarstellung ist stufenweise möglich. Diagramme können auch automatisch an die Bildschirmgröße angepasst werden. Es ist möglich die Größe der verschiedenen Diagrammelemente per Maus zu verändern. Die minimale Größe ist abhängig von der enthaltenen Beschreibung. Zeilenumbrüche sind nicht erlaubt.

Assoziationen können individuell angeordnet werden um Überschneidungen mit anderen Elementen zu vermeiden. Es ist nicht möglich die Strichstärke und Linienart einzustellen.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-	+	+	+	+	+	-	+	+	+	o	-	+	-	+

f) Graphikeditor / Bedienung

Das Einzeichnen von Symbolen erfolgt über die Werkzeugbox und das Aufziehen der Elemente mit der Maus. Wird ein Element zum Diagramm hinzugeführt (z.B. eine Klasse), kann diese sofort bezeichnet und bearbeitet werden. Will der Benutzer nacheinander mehrere gleiche Elemente in ein Diagramm hinzufügen, kann dies durch ein Doppelklick auf das gewünschte Element im Werkzeugkasten erfolgen. Das gewünschte Werkzeug bleibt dadurch aktiv. Das Verschieben der Elemente erfolgt mit der Maus per drag-and-drop. Der Zeichenbereich vergrößert sich automatisch und in einer Bildschirmseite kann horizontal und vertikal gescrollt werden. Mehrere Elemente können per Maus ausgewählt werden und gemeinsam bearbeitet werden. Hinzugefügte Elemente erscheinen dort, wo der Benutzer mit der Maus auf die Zeichenfläche klickt. Es ist immer möglich, das Diagramm händisch zu manipulieren. Eine irrtümliche Zeichenaktion kann nicht abgebrochen werden. Das Entfernen von falschen Informationen erfolgt durch die Entfernen-Taste, nachdem das zu entfernende Element zuvor mit der Maus markiert wurde. Die Arbeit wird mittels Undo- und Redo-Funktionen erheblich erleichtert. Der Diagrammeditor stellt die üblichen Zwischenablagefunktionen wie Cut, Copy und Paste bereit. Es ist möglich Elemente aus dem Diagrammen zu schneiden, kopieren und in ein anderes Diagramm einzufügen.

1	2	3	4	5	6	7	8	9	10	11	12	13	14
+	+	+	+	+	+	+	+	-	+	+	+	+	+

g) Performance

Bei der Performance von StarUML gibt es keine Beanstandungen. Das Zoomen und der Bildaufbau erfolgen zügig. Auch die Performance der automatischen Codegenerierung ist zufrieden stellend.

1	2	3	4
+	+	+	+

h) Adaptivität

Eine Anpassung von StarUML an die Bedürfnisse bzw. Lernfortschritts der Benutzer ist nicht möglich. Für die unterstützten Programmiersprachen ist es aber möglich, mit Profilen zu arbeiten.

Die Symbolleisten können individuell angepasst werden, es ist aber nicht möglich, neue Funktionstastenbelegungen zu definieren. Auch das Verändern der Menüs ist nicht möglich.

Für StarUML kann der Quellcode von der Entwicklerhomepage geladen und angepasst werden, da es sich hierbei um ein Open-Source Produkt handelt.

1	2	3	4	5	6
-	o	-	-	-	+

5.2 Poseidon for UML

5.2.1. Methodenbasierte Kriterien

a) Modellierung der Objektstruktur – Objektstatik

Die Speicherung von neuen Objekten und Informationen werden im Data-Dictionary automatisiert abgelegt. Beim Anlegen von neuen Elementen werden diese aber nicht auf Namensgleichheiten überprüft. Alle Informationen sind im Diagrammfenster in Form eines Baummenüs dargestellt. Von dort aus können diese auch manipuliert werden. Von Attributen können der Name, die Sichtbarkeit, der Typ und der Startwert angegeben werden. Für Methoden können noch Einschränkungen definiert werden.

1	2	3	4	5
+	+	+	+	+

b) Modellierung der Systemstruktur – Systemstatik

Rekursive Beziehungen können mit Poseidon for UML modelliert werden. Einmal erfasste Beziehungen können nachträglich mittels drag-and-drop verändert werden. Object Constraint Language (OCL) – Annotationen werden von diesem Tool unterstützt. Die Einteilung von Klassen erfolgt mittels Packages.

1	2	3	4
+	+	o	+

c) Dynamisches Objektverhalten – Objektdynamik

Es können für jede Klasse Zustandsdiagramme erstellt werden, die umkehrbar eindeutig mit der jeweiligen Klasse verbunden sind. Ebenso besitzt Poseidon for UML die Möglichkeit Bedingungen für Zustandsübergänge zu definieren. Aktivitäten und Ereignisse können im Sinne einer genaueren Beschreibung verfeinert werden. Sequenzdiagramme nehmen Bezug auf Methoden bzw. Attribute des statischen Objektmodells.

1	2	3	4	5	6	7
+	+	+	+	o	-	+

d) Zusammenhang zwischen den Modellen

Jedes Zustandsdiagramm ist umkehrbar eindeutig mit einem Objekt bzw. einer Klasse verbunden. Im Browser werden Zustandsdiagramme nicht unter der jewei-

ligen Klasse angezeigt. Das dynamische Verhalten einer Klasse wird nicht weitervererbt.

1	2	3	4
+	-	+	-

e) Abstraktionsebenen / Sichten

Bei Poseidon for UML können die Sichtbarkeit, Stereotype, Attribute oder die Methoden ein- bzw. ausgeblendet werden. Die Anzeige von Parametern kann im Programmfenster Einstellungen angepasst werden. Objektkomponenten können für eine Variation der Genauigkeit der Darstellung verändert werden.

1	2	3	4
+	o	+	-

f) Vorgangsreihenfolge bei der Entwicklung

Die Verwaltung des Repositorys ist unabhängig von den Diagrammsichten und deshalb brauch sich der Benutzer nicht an bestimmte Vorgehensmodelle halten. Die Daten können direkt im Repository-Browser eingetragen oder manipuliert werden und später in Diagrammen verwendet werden. Umgekehrt funktioniert dieses Prinzip auch.

1	2	3
+	+	+

g) Unterstützung bei der Methodenanwendung

Das Tool besitzt keine mitgelieferte UML-Hilfe. Auf der Herstellerseite findet man aber genügend Lernhilfen und Informationen zu UML. Einige Notationselemente unterscheiden sich geringfügig von UML-Standard. Eine Konsistenz- und Integritätsprüfung ist nicht vorhanden.

1	2	3
o	o	-

5.2.2. Funktionale Kriterien

a) Datenhaltung

Das Speichern der Modellierungsinformationen erfolgt im *Modell-Explore*r. In Form eines Baummenüs kann auf die Daten zugegriffen werden. Werden Informationen aus den Diagrammen entfernt, löschen sich auch dieselben Informationen im Modell Explorer automatisch. Das Manipulieren von Daten ist auch direkt im Repository möglich. Die Daten werden im ZUML-Format gespeichert. Inwieweit diese dann von anderen UML-Werkzeugen bearbeitet werden können ist nicht bekannt. Auch eine Ausgabe als XMI-File ist möglich.

Importiert werden können nur Java-Quellcode Dateien. Auf Diagramme anderer UML-Tools kann nicht zugegriffen werden.

1	2	3	4	5
+	+	+	+	o

b) Browsing-Eigenschaften

In Poseidon for UML existieren zwei Suchfunktionen. Mit einer kann nach Elementen gesucht werden und mit der Zweiten kann nach Diagrammen gesucht werden. Ein Pattern Matching ist nicht integriert. Es kann auch nach Elementen per Anfangsbuchstabe gesucht werden. Die Ergebnisliste berechnet sich dynamisch. Filter können nicht gesetzt werden, ebenso werden keine weitere Informationen über Suchergebnisse in der Trefferliste angezeigt.

Änderungen von Informationen sind auch direkt im Repository möglich. Diese werden auch automatisch in die Diagramme übernommen. Eine textuelle Darstellung von Klassenhierarchien zur besseren Übersicht ist nicht möglich.

1	2	3	4	5	6	7	8	9	10	11
+	-	+	-	-	-	+	-	-	-	+

c) Navigation im Systemmodell

Um zwischen den einzelnen Diagrammen navigieren zu können muss der *Diagramm Explorer* geöffnet sein. Über ein Baummenü sind alle Diagramme erreichbar. Es ist nicht möglich, über ein Zustandsdiagramm zum entsprechenden Klassendiagramm zu wechseln. Informationen zu den verschiedenen Elementen werden im Eigenschaften-Fenster angeführt, das vorher geöffnet sein muss.

1	2	3
+	-	o

d) Konsistenzprüfung / Integrität

Assoziationen ins Leere sind nicht möglich. Sobald man eine Assoziation, die ins Leere gehen soll anlegt, wird automatisch eine neue Klasse ans Ende der neuen Assoziation angelegt. Es ist möglich, rekursive Vererbungen zu erstellen. Das System erzeugt keine Fehlermeldung. Ebenso können Modellelemente mit gleicher Bezeichnung angelegt werden. Das System prüft nicht auf Ähnlichkeiten zwischen Bezeichnungen. Eine Konsistenzprüfung kann nicht automatisch durchgeführt werden. Referentielle Integrität wird nicht unterstützt, d.h. Werte von Fremdschlüsseln können auch leer sein. Werden Teile aus einem Diagramm gelöscht, werden diese auch automatisch aus der Datenbasis gelöscht.

1	2	3	4	5	6
+	-	-	-	-	-

e) Report-Ausgabe

In Poseidon for UML ist eine Report-Ausgabe nicht vorhanden.

1	2	3	4	5	6	7	8
-	-	-	-	-	-	-	-

f) Datenbankschema-Ausgabe

Es ist nicht möglich, ein Datenbankschema für relationale und objektorientierte Datenbanken auf Basis der Informationen des Objektmodells zu generieren.

1	2	3	4	5
-	-	-	-	-

g) Diagramm-Ausgabe

Die Diagrammausgabe von Poseidon for UML ist recht schlicht gehalten. Es können die gewünschten Druckgröße und die Diagrammausrichtung jedoch kein Abstraktionsniveau eingestellt werden. Die Ausgabe entspricht der Bildschirmansicht.

1	2	3	4	5
+	-	+	-	+

h) Offenheit – Import / Export

Poseidon for UML kann mit einigen weiteren Plug-Ins erweitert werden. Diese können gegen ein Entgelt vom Hersteller bezogen werden. Es ist möglich in Poseidon for UML die kostenlose Java Entwicklungsumgebung Eclipse zu integrieren. Das Tool schöpft die von der Plattform gebotenen Möglichkeiten des Datenaustauschs (über die Zwischenablage) aus. Es ist nicht möglich, Grafik- und Textformate in den Zeicheneditor einzulesen. Tools von Drittherstellern können Daten, die im XMI-Format exportiert wurden, lesen und bearbeiten.

1	2	3	4	5
+	+	-	-	+

i) Codegenerierung

Die automatische Codegenerierung wird nur für Java-Source-Dateien unterstützt. Für jede im Modell vorkommende Klasse wird eine entsprechende Codierung vorgenommen. Attribute und Typen werden übernommen. Methodengerüste und eventuelle Zugriffsbeschränkungen werden automatisch erstellt. Das Ausmaß der Codegenerierung lässt sich nicht bestimmen.

1	2	3	4	5	6	7	8	9	10	11
+	+	+	+	-	+	+	+	-	-	-

j) Reverse-Engineering

Reverse Engineering ist für Java-Dateien möglich. Diagramme werden aus Sourcecode-Dateien ohne Probleme erstellt. Die Kommentare in der Quelldatei werden auch übernommen und müssen manuell aus dem *Model Explorer* gelöscht werden. Es werden automatisch Packages angelegt. Auch Code, der nicht von Poseidon for UML erstellt wurde, kann eingelesen werden. Auch Code innerhalb von Methoden kann eingelesen und in einem internen Editor bearbeitet werden.

1	2	3
+	+	+

k) Test / Verifikation

Eine Möglichkeit zum Testen bzw. zur Simulation auf Modellebene wird von Poseidon for UML nicht unterstützt.

1	2
-	-

l) Maßzahlen und Statistiken

Maßzahlen und Statistiken kann Poseidon for UML nicht generieren. Es können keine Aussagen über Anzahl der Klassen, Attributen, Methoden, usw. gemacht werden.

1	2	3
-	-	-

m) Multi-User – Eigenschaften

Mehrere User könnten nur auf die gespeicherten Daten zugreifen, wenn diese von einem Server verwaltet werden. Eigentliche Multi-User Features werden nicht angeboten. Ein Userkonzept ist nicht vorhanden. Ein Locking auf Klassen oder Modellebene ist nicht möglich.

1	2	3	3
o	-	-	-

n) Security

Der Benutzer braucht sich nach dem Programmstart nicht anzumelden. Eine solche Sicherheitsfunktion ist nicht implementiert. Auch das Protokollieren von Änderungen durch den Benutzer ist nicht umgesetzt. Eine integrierte Datensicherung und eine Recovery-Funktion für die Integrität des Programms und der Daten sind nicht vorhanden. Bei Bedienungsfehlern werden dem Benutzer entsprechende Meldungen geliefert und das UML-Werkzeug bleibt in einem definierten Zustand.

Nach dem Löschen von Modellierungsinformationen muss der Benutzer diese Aktion noch einmal bestätigen. Dadurch soll ein unbeabsichtigtes Löschen von Da-

ten verhindert werden. Durch die Undo-Funktion kann auch ein Löschen von Informationen rückgängig gemacht werden.

1	2	3	4	5	6	7	8	9
-	-	-	-	-	-	+	+	+

o) Multi-Projekt – Eigenschaften / Versionsmanagement

Eine zeitlich parallele Abwicklung von mehreren Projekten ist nicht möglich. Es kann nur ein Projekt zu einem Zeitpunkt geöffnet sein, außer man startet die Applikation öfters. Ein Austausch von Repository-Daten ist nur durch die Import/Export-Funktion möglich. Das Angeben von Referenzen ist nicht möglich. Ein Versionsmanagement wird nicht unterstützt.

1	2	3	4	5
o	o	-	-	-

5.2.3. Ergonomie

a) Benutzeroberfläche

Das Aussehen von Poseidon for UML besitzt keine Ähnlichkeit mit bekannten Windows-Programmen. Es erinnert an typische Java-Programme, da es auch in dieser Programmiersprache geschrieben ist. Dennoch ist das GUI übersichtlich gestaltet. Die Werkzeuge sind übersichtlich in Baumstrukturen angeordnet. Die Dialoge und Eingabemasken sind einfach zu bedienen. Die Namensgebung der Menüs und Funktionen ist klar definiert und ist selbstbeschreibend, einheitlich und widerspruchsfrei. Deaktivierte Funktionen sind grau hinterlegt. Bei länger dauernden Aktionen bekommt der Benutzer vom System nicht immer das gewünschte Feedback durch Fortschrittsbalken oder Sanduhren. Um genügend Platz für die Erstellung von Diagrammen am Bildschirm zu haben, können Teilfenster zugeklappt oder geschlossen werden.

1	2	3	4	5	6	7
+	+	+	+	+	o	+

b) Tool-Dokumentation

Mit dem Herunterladen von Poseidon for UML wird auf dem Rechner auch ein Benutzerhandbuch installiert, das vom Programm aus geöffnet werden kann. Das Benutzerhandbuch ist in englischer Sprache verfasst und als HTML-Dokument verfügbar. Mit Hilfe von Screenshots werden die einzelnen Funktionen ausführlich beschrieben und dargestellt. Außerdem gibt es in diesem Dokument auch eine schriftliche Anleitung zur Installation, sowie eine „Erste-Schritte“ Anleitung für un-

erfahrene Benutzer. Weitere Hilfestellungen findet der Benutzer auf der Herstellerhomepage in Form von PDF-Dokumenten, die man auf seinen Rechner herunterladen kann.

Ebenso ist im Tool schon eine Beispielanwendung integriert, die sich auch zum Selbststudium eignet.

1	2	3	4	5	6	7	8	9	10	11	12
+	+	-	-	+	o	-	o	+	+	+	+

c) Usability

Das Tool ermöglicht ein rasches und effizientes Arbeiten. Die übersichtlich angeordneten Werkzeuge und die vorhandenen Shortcuts lassen den Anwender schnell auf die von ihm gewünschten Funktionen zugreifen. Die Systemausgaben in Form von Fehlermeldungen sind klar und verständlich. Das Belegen von Funktionstasten ist leider nicht möglich. Die Kontextmenüs sind übersichtlich und funktionell. Fehlerhafte Eingaben werden vom System abgefangen und mit dem entsprechenden Feedback ausgegeben. Die vorhandenen Symbolleisten sind übersichtlich angeordnet, jedoch kann der Benutzer die Symbolleisten nicht individuell an seine Bedürfnisse anpassen. Längere Systemoperationen (wie z.B. Laden oder Speichern von Projekten) werden mittels Fortschrittsanzeigen weiteren Informationen angezeigt. Zur verbesserten Übersicht großer Diagramme verfügt Poseidon for UML ein Übersichtsfenster, mit dem man auch in das Modell Ein- bzw. Auszoomen kann. Verschiedenen Diagrammtypen können gleichzeitig, aber nicht nebeneinander betrachtet werden. Eine Mehrfachauswahl von Elementen ist mit gedrückter Shift-Taste möglich. Poseidon for UML ist in englischer und in deutscher Sprache verfügbar.

1	2	3	4	5	6	7	8	9	10	11
o	o	+	-	+	+	o	+	o	+	+

d) Hilfefunktionen

Die Hilfefunktion ist nur über das Menü erreichbar und kann nicht über die Funktionstaste F1 aufgerufen werden. Es kann nur das Handbuch als Hilfe verwendet werden, das im HTML-Format vorliegt. Ein Suchen nach Themengebieten ist nicht möglich. Hot-Help für die verschiedenen Symbole ist verfügbar.

1	2	3	4
o	+	-	+

e) Graphikeditor / Darstellung

Es ist bei Poseidon for UML nicht möglich, die Symbole in einem bestimmten Umfang anzupassen. Es existieren limitierte Formen, um eigene Diagramme bzw. Zeichnungen zu erstellen.

Die am Bildschirm dargestellten Diagramme entsprechen dem späteren Ausdruck. Die Farben der verschiedenen Notationselemente können nicht für jedes Element einzeln verändert werden. Elemente können horizontal bzw. vertikal au-

tomatisch ausgerichtet werden. Poseidon for UML besitzt ein stufenloses Zoom, das bequem über das Übersichtsfenster per Schieberegler gesetzt werden kann. Symbole und Texte können nicht rotiert werden.

Die Elemente werden automatisch in der optimalen Größe am Bildschirm angezeigt. Beim Speichern und erneuten Laden von Diagrammen wird das Layout beibehalten.

Die Schriftgröße kann verändert werden. Assoziationen können durch nachträglich festgelegte Eckpunkte verändert werden. Die Strichstärke und Linienart kann man nicht verändern. Ein Refresh ist nicht vorhanden.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-	o	+	o	+	+	-	+	+	o	+	-	+	-	-

f) Graphikeditor / Bedienung

Bei Poseidon for UML hat der Benutzer zwei Möglichkeiten, Diagramme zu erstellen. Die erste Möglichkeit ist die Auswahl des jeweiligen Werkzeugs und das Aufziehen der Elemente mit der Maus auf der Zeichenfläche. Die zweite Möglichkeit ist das Zeichnen von Diagrammen, ohne immer wieder das gewünschte Werkzeug auszuwählen. Hierfür werden die zur Verfügung stehenden Werkzeuge in Form von Icons auf der Zeichenfläche eingeblendet. Diese Art der Erstellung von Diagrammen ist schneller, da sich der Benutzer dadurch lange Mauswege und eine Menge von Mausklicks erspart. Bevor man aber die Vorteile dieses Bedienkonzeptes effizient nutzen kann, benötigt der Benutzer einige Kenntnisse über den Umgang mit Poseidon for UML. Nach dem Anlegen der Notationselemente können diese sofort textuell beschrieben werden. Eine Funktion zum vielfachen Einfügen derselben Elemente hintereinander ist auch vorhanden. Die Bedienung erfolg intuitiv. Darunter fällt auch die Manipulation der Diagrammelemente, wie z.B. das Verschieben der Elemente.

Objekte können automatisch horizontal bzw. vertikal an Gitterpunkten ausgerichtet werden. Um mehrere Symbole zu bearbeiten, können diese mit der Maus umrahmt werden. Mittels Undo- und Redo- Funktionen können Bearbeitungsschritte rückgängig bzw. wiederholt werden. Auch die Funktionen der Zwischenablage werden hier unterstützt (Copy/Cut und Paste). Durch die Entfernen-Taste können Symbole aus den Diagrammen wieder entfernt werden.

1	2	3	4	5	6	7	8	9	10	11	12	13	14
+	+	+	+	+	+	+	+	-	+	+	+	+	+

g) Performance

Gegenüber anderen UML-Werkzeugen erfolgt Programmstart etwas schleppend. Auch das Arbeiten mit großen Modellen wird nach einiger Zeit ein wenig zeitversetzt. Bei der automatischen Codegenerierung und der Suche nach Daten gibt es keine Beanstandungen.

1	2	3	4
o	o	+	+

h) Adaptivität

Bezüglich der Adaptivität schneidet Poseidon for UML nicht gut ab. Dieses Produkt stellt keine Funktionen zur Verfügung, entsprechend des Lernfortschritts des Anwenders die Menü- und Symbolleisteneinträge zu verändern und in ihrem Umfang anzupassen.

Auch der Quellcode des Programms wird nicht mitgeliefert.

1	2	3	4	5	6
-	-	-	-	-	-

5.3 Objecteering 6.1 Free Edition

5.3.1. Methodenbasierte Kriterien

a) Modellierung der Objektstruktur – Objektstatik

Objecteering 6.1 Free Edition hat kein gemeinsames Object-Dictionary. Das Anlegen von gleichen Komponenten mit gleichen Elementen mit gleichen Namen ist möglich, man erhält jedoch in einem Fenster eine Fehlerbeschreibung. Alle erstellten Elemente sind im Modell-Explorer in Form einer Baumstruktur dargestellt. Eine Manipulation der Elemente ist sowohl von der Baumstruktur aus wie auch direkt im Diagramm möglich. Bei Attributen können Name, die Sichtbarkeit, Typ und Constraints angegeben werden. Für Methoden können noch Einschränkungen definiert werden. Für Methoden können zusätzlich Ein- und Ausgangsparameter festgelegt werden.

1	2	3	4	5
+	+	+	+	o

b) Modellierung der Systemstruktur – Systemstatik

Rekursive Beziehungen können mit Objecteering Free Edition modelliert werden. Einmal erfasste Beziehungen können nachträglich mittels drag-and-drop verändert werden. Dies gilt auch, wenn sich die Beziehung auf eine andere Klasse beziehen soll. Object Constraint Language (OCL) – Annotationen werden von Objecteering 6 Free Edition unterstützt. Die Einteilung in Klassen ist mittels Packages möglich.

1	2	3	4
+	+	o	+

c) Dynamisches Objektverhalten – Objektdynamik

Es können für jede Klasse Zustandsdiagramme erstellt werden, die umkehrbar eindeutig mit der jeweiligen Klasse verbunden sind. Es besteht auch die Möglichkeit, Bedingungen für Zustandsübergänge zu definieren. Aktivitäten und Ereignisse können durch eine genauere Beschreibung verfeinert werden. Geschachtelte Zustandsdiagramme sind nicht möglich. Generalisierungen von Ereignissen kön-

nen nicht modelliert werden. Sequenzdiagramme nehmen Bezug auf Methoden bzw. Attribute des statischen Objektmodells.

1	2	3	4	5	6	7
+	+	+	+	-	-	-

d) Zusammenhang zwischen den Modellen

Jedes Zustandsdiagramm ist umkehrbar eindeutig mit einem Objekt bzw. einer Klasse verbunden. Im Browser werden Zustandsdiagramme nicht unter der jeweiligen Klasse angezeigt. Das dynamische Verhalten einer Klasse wird nicht weitervererbt.

1	2	3	4
+	-	+	-

e) Abstraktionsebenen / Sichten

Bei Objecteering 6 Free Edition können Sichtbarkeit, Parameter, Signatur und Typen von Attributen und Methoden ein- bzw. ausgeblendet werden. Man kann für jede Klasse einzeln verschiedene Subelemente (alle Attribute, alle Methoden,…) oder auch alle Subelemente gemeinsam ein- bzw. ausblenden. Außerdem ist es möglich, einzelne Attribute oder Methoden auszublenden.

1	2	3	4
+	+	+	-

f) Vorgangsreihenfolge bei der Entwicklung

Der Benutzer muss sich nicht an bestimmte Vorgehensmodelle halten. Elemente und Daten können eingegeben und manipuliert werden. Man kann zum Beispiel eine Klasse näher beschreiben, ohne sie vorher zu benennen. Man wird jedoch im UML-Audit-Bereich auf den Fehler hingewiesen. Das gilt auch für andere Elemente. Eine rein textuelle Erfassung von Modellelementen ist nicht möglich.

1	2	3
+	-	o

g) Unterstützung bei der Methodenanwendung

Bei der Installation von Objecteering 6 Free Edition werden 4 verschiedene Dokumentationen und Hilfen mitkopiert. Man kann diese vom Programm aus oder über das Startmenü unter Objecteering 6 Free Edition / help aufrufen. Die Hilfe enthält unter anderem Bescheibungen zu den unterschiedlichen Diagrammtypen inklusive grafischen Beispielen. Einige Notationselemente unterscheiden sich geringfügig vom UML-Standard.

Die Konsistenz und Integrität wird laufend geprüft (UMLAudit-Fenster). Durch Doppelklick auf das Problem erhält man eine nähere Problembeschreibung.

1	2	3
+	o	+

5.3.2. Funktionale Kriterien

a) Datenhaltung

Das Speichern der Modellierungsinformationen erfolgt im *Modell-Explore*r. In Form eines Baummenüs kann auf die Daten zugegriffen werden. Werden Informationen aus den Diagrammen über das Kontextmenü und „Delete“ entfernt, löschen sich auch dieselben Informationen im Modell Explorer automatisch. Drückt man nur die „Entf“-Taste, bleiben die Daten im Modell-Explorer erhalten. Das Manipulieren von Daten ist auch direkt im Modell-Explorer möglich. Man kann nur Objecteering-Projekte importieren.

1	2	3	4	5
+	+	+	-	-

b) Browsing-Eigenschaften

Objecteering 6 Free Edition enthält viele Möglichkeiten, um eine Elementsuche einzugrenzen. Man kann nach beinhalteten Buchstaben, Anfangsbuchstaben, Endbuchstaben und exakter Übereinstimmung suchen. Außerdem lassen sich die Suchbereiche auf Packages einschränken. Die Suche nach Anfangsbuchstaben bietet keine Auswahl der zutreffenden Elemente an. Man kann die Suche zusätzlich auf einen bestimmten Elementtyp beschränken. Auch der Gebrauch eines Filters ist möglich. Zusatzinformationen zu gefundenen Elementen werden nicht angezeigt.

Änderungen von Informationen sind auch direkt im Modell-Explorer möglich. Diese werden auch automatisch in die Diagramme übernommen. Eine textuelle Darstellung von Klassenhierarchien zur besseren Übersicht ist nicht möglich.

1	2	3	4	5	6	7	8	9	10	11
+	-	o	+	-	-	+	-	-	-	+

c) Navigation im Systemmodell

Über den Explorer kann man recht einfach zwischen den Diagrammen navigieren. Über das Baummenü sind alle Diagramme erreichbar. Es ist nicht möglich über ein Zustandsdiagramm zum entsprechenden Klassendiagramm zu wechseln.

Informationen zu den verschiedenen Elementen werden im Eigenschaften-Fenster angeführt, das mittels Doppelklick aufgerufen werden kann.

1	2	3
+	-	+

d) Konsistenzprüfung / Integrität

Assoziationen ins Leere sind nicht möglich. Es ist möglich rekursive Vererbungen zu erstellen. Das System erzeugt keine Fehlermeldung. Ebenso können Modellelemente mit gleicher Bezeichnung angelegt werden. Das System prüft nicht auf Ähnlichkeiten zwischen Bezeichnungen. Eine Konsistenzprüfung kann nicht automatisch durchgeführt werden. Entfernt man Elemente mittels „Entf“-Taste aus einem Diagramm, so bleibt das Element in der Datenbasis trotzdem bestehen. Wird es jedoch über das Kontextmenü und „Delete“ entfernt, so verschwindet es auch aus der Datenbasis.

1	2	3	4	5	6
+	-	-	-	+	o

e) Report-Ausgabe

Objecteering 6 Free Edition bietet keine Funktion zur Ausgabe von Reports.

1	2	3	4	5	6	7	8
-	-	-	-	-	-	-	-

f) Datenbankschema-Ausgabe

Es ist nicht möglich, ein Datenbankschema für relationale und objektorientierte Datenbanken auf Basis der Informationen des Objektmodells zu generieren.

1	2	3	4	5
-	-	-	-	-

g) Diagramm-Ausgabe

Die Diagrammausgabe von Objecteering 6 Free Edition ist extrem schlicht. Es gibt keine Ausgabeoptionen (außer den Einstellungen, die die Druckersoftware selbst bietet). Die Ausgabe enthält zusätzlich zur Bildschirmansicht Objecteering-Wasserzeichen.

1	2	3	4	5
+	-	o	-	o

h) Offenheit – Import / Export

Es wird ein Add-On angeboten, das es erlaubt Objecteering in ModelBus einzubinden. ModelBus liefert die Möglichkeit, transparent auf Modellierungsservices zuzugreifen. Das Add-On kann ebenfalls kostenlos von der Homepage des Herstellers heruntergeladen werden.

1	2	3	4	5
-	-	-	-	+

i) Codegenerierung

Die automatische Codegenerierung wird für Java, C++ und Fortran unterstützt. Man muss dafür aber erst die entsprechende MDAC-Komponente über das Menü Tools / Deploy MDAC hinzufügen. Am Besten funktioniert die Codegenerierung in Java. Für jede im Modell vorkommende Klasse wird eine entsprechende Codierung vorgenommen. Attribute und Typen werden übernommen. Methodengerüste und eventuelle Zugriffsbeschränkungen werden automatisch erstellt. Das Ausmaß der Codegenerierung lässt sich nicht bestimmen. Will man ein File neu generieren, dessen Code händisch erweitert wurde, wird man gefragt, ob man das File behalten oder neu generieren will. Eine einfache Erweiterung ist nicht möglich.

1	2	3	4	5	6	7	8	9	10	11
+	+	+	+	-	+	+	+	-	-	-

j) Reverse-Engineering

Reverse Engineering ist für Java, C++ und C# möglich. Es werden nur Attribute und Methodensignaturen übernommen. Es können auch von anderen Tools generierte Files eingelesen werden. Der Methodeninhalt und die Dokumentationen können eingelesen und bearbeitet werden.

1	2	3
+	+	+

k) Test / Verifikation

Objecteering 6 Free Edition bietet keine Möglichkeiten für Test und Simulation.

1	2
-	-

l) Maßzahlen und Statistiken

Maßzahlen und Statistiken können von Objecteering 6 Free Edition nicht generiert werden. Es können keine Aussagen über Anzahl der Klassen, Attributen, Methoden, usw. gemacht werden.

1	2	3
-	-	-

m) Multi-User – Eigenschaften

Multi-User-Features werden von der Objecteering 6 Free Edition nicht unterstützt.

1	2	3	3
-	-	-	-

n) Security

Da von der Objecteering 6 Free Edition keine Multi-User-Features unterstützt werden, gibt es keine Anmeldung mittels Benutzername und Kennwort. Es können auch keine Zugriffsrechte vergeben werden. Es gibt weder ein Logging-Protokoll noch eine automatisierbare Datensicherung.

Das Programm gibt bei Bedienfehlern Meldungen aus und bleibt im definierten Zustand. Unbeabsichtigtes Löschen wird durch Bestätigungsabfragen verhindert und es existiert eine Undo-Funktion.

1	2	3	4	5	6	7	8	9
-	-	-	-	-	-	+	+	+

o) Multi-Projekt – Eigenschaften / Versionsmanagement

Eine zeitlich parallele Abwicklung von mehreren Projekten ist nicht möglich. Es kann nur ein Projekt zu einem Zeitpunkt geöffnet sein, außer man startet die Applikation mehrmals. Ein Austausch von Repository-Daten ist nur durch die Import-Funktion möglich. Referenzen sind nicht möglich. Ein Versionsmanagement wird nicht unterstützt.

1	2	3	4	5
o	o	-	-	-

5.3.3. Ergonomie

a) Benutzeroberfläche

Der Aufbau des GUI erinnert eher an (JAVA)-Programmierumgebungen als an übliche Windows-Programme. Die Menü- und Symbolleisten sind Windows-orientiert. Dennoch ist das GUI im Allgemeinen übersichtlich gestaltet. Die Werk-

zeuge sind übersichtlich in Baumstrukturen angeordnet. Die Dialoge und Eingabemasken sind zum Teil etwas gewöhnungsbedürftig. Zum Beispiel muss man zuerst eine State-Machine anlegen und kann erst dann ein Zustandsdiagramm anlegen. Die Namensgebung der Menüs und Funktionen ist klar definiert und ist selbsterklärend, einheitlich und widerspruchsfrei. Deaktivierte Funktionen sind nicht explizit hervorgehoben. Will man jedoch beispielsweise etwas rückgängig machen, wenn nichts mehr da ist, was rückgängig gemacht werden könnte, erhält man eine entsprechende Meldung in der Konsole. Bei länger dauernden Aktionen (zB. Laden einer MDA-Komponente) bekommt der Benutzer vom System nicht immer das gewünschte Feedback durch Fortschrittsbalken oder Sanduhren. Um genügend Platz für die Erstellung von Diagrammen am Bildschirm zu haben, können Teilfenster zugeklappt oder geschlossen werden.

1	2	3	4	5	6	7
+	+	o	+	-	o	+

b) Tool-Dokumentation

Bei der Installation von Objecteering 6 Free Edition werden Benutzerhandbücher mitinstalliert, die vom Programm aus geöffnet werden können. Das Benutzerhandbuch ist in englischer Sprache verfasst und als Html-Dokument verfügbar. Mit Hilfe von Screenshots werden die einzelnen Funktionen ausführlich beschrieben und dargestellt. Außerdem gibt es in diesem Dokument auch eine schriftliche Anleitung zur Installation, sowie eine „Erste-Schritte“ Anleitung für unerfahrene Benutzer. Gedruckte Dokumentationen oder pdf-Dokumente existieren nicht.

1	2	3	4	5	6	7	8	9	10	11	12
o	+	-	-	-	o	-	o	+	+	-	+

c) Usability

Länger andauernde Aktionen können nicht beliebig abgebrochen werden. Fehlermeldungen werden in Konsole ausgeben. Die Meldung enthält jedoch keine Information darüber warum eine Aktion nicht durchgeführt werden kann oder wie man das Problem lösen könnte. Tastatur-Shortcuts für Menüfunktionen sind vorhanden. Die Zuweisung von Befehlen an die Funktionstasten ist nicht möglich. Ein Kontextmenü ist vorhanden. Die Symbolleisten enthalten keine Symbole, die man für die Modellierung selbst verwenden könnte. Sie enthalten nur allgemeine Befehle wie Suchen, Drucken, Speichern,… Die Symbolleisten können nicht individuell verändert werden. Objecteering 6 Free Edition enthält zwar eine Statuszeile. Diese enthält jedoch keine Informationen. Sie zwigt immer „Ready“, auch wenn das Programm gerade beschäftigt ist. Eine Mehrfachauswahl von Elementen ist mit gedrückter Strg-Taste möglich. Eine deutschsprachige Version der Objecteering 6 Free Edition ist nicht verfügbar.

1	2	3	4	5	6	7	8	9	10	11
-	-	+	-	+	+	-	-	-	+	o

d) Hilfefunktionen

Die Hilfefunktion ist nur über das Menü erreichbar und kann nicht über die Funktionstaste F1 aufgerufen werden. Es kann aber nur das Handbuch als Hilfe verwendet werden, das im Html-Format vorliegt. Ein Suchen nach Themengebieten ist nicht möglich. Man kann sich nur von Link zu Link durchklicken.

1	2	3	4
-	+	-	-

e) Graphikeditor / Darstellung

Es ist bei der Objecteering 6 Free Edition nicht möglich, Symbole anzupassen oder gar eigene zu definieren. Man kann auch keine Diagramme aus gängigen Umrissformen erstellen. Abgesehen vom Wasserzeichen des Programms entsprechen die am Bildschirm dargestellten Diagramme dem späteren Ausdruck. Die Farben der einzelnen Notationselemente können individuell verändert werden. Elemente können horizontal bzw. vertikal automatisch ausgerichtet werden. Das Zoomen ist mittels Mausrad möglich (Scrolltasten eines Laptops reichen nicht aus). Symbole und Texte können nicht rotiert werden. Es gibt eine Funktion (Symbol) für ein automatisches Diagramm-Layout. Beim Speichern und erneutem Laden von Diagrammen wird das Layout beibehalten. Die Schriftgröße kann über die Grafikoptionen verändert werden und ist gut lesbar.

Assoziationen können durch nachträglich festgelegte Eckpunkte verändert werden. Die Strichstärke und Linienart kann man nicht verändern. Ein Refresh ist nicht vorhanden.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-	-	+	+	+	+	-	+	+	+	+	-	+	+	-

f) Graphikeditor / Bedienung

Der Benutzer kann ein Diagramm anlegen, indem er das gewünschte Diagramm aus der Symbolleiste am linken Fensterrand auswählt. Die für das jeweilige Diagramm zur Verfügung stehenden Werkzeuge werden dann am linken Rand des Zeichenbereichs in Form von Icons eingeblendet. Einige Symbole für Diagrammtypen sind etwas versteckt. Um zum Beispiel ein Activity-Diagramm anzulegen, muss man zuerst einen Activity-Graph anlegen. Erst von diesem ausgehend kann man ein Activity-Diagramm anlegen. Nach dem Anlegen der Notationselemente können diese sofort textuell beschrieben werden. Eine Funktion zum vielfachen Einfügen der desselben Elemente hintereinander ist nicht vorhanden. Die Bedienung erfolgt größtenteils intuitiv. Darunter fällt auch die Manipulation der Diagrammelemente, wie z.B. das Verschieben der Elemente.

Objekte können automatisch horizontal bzw. vertikal an Gitterpunkte ausgerichtet werden. Das Gitternetz ist nicht frei definierbar. Die Anwendung von Funktionen und Manipulationen für mehrere Elemente gleichzeitig durch Gruppierung ist nicht möglich. Mittels Undo- und Redo- Funktionen können Bearbeitungsschritte rückgängig gemacht bzw. wiederholt werden. Copy/Paste sind im Kontextmenü nicht vorhanden. Auch die Verwendung von Copy/Paste aus der Menüleiste bringt nicht den erwarteten Erfolg. Durch die „Entf“-Taste können Symbole aus den Dia-

grammen wieder entfernt werden. Allerdings muss das Element aus der Baumstruktur im UML-Explorer entfernt werden, damit es wirklich verschwindet.

1	2	3	4	5	6	7	8	9	10	11	12	13	14
+	+	-	+	+	o	+	-	-	+	+	+	-	+

g) Performance

Das Programm ist schnell geladen und erlaubt zügiges und unterbrechungsfreies Arbeiten. Auch automatisch generierter Code und Suchergebnisse stehen rasch zur Verfügung und können verwendet werden.

1	2	3	4
+	+	+	+

h) Adaptivität

Objecteering 6 Free Edition bietet keine Adaptivität. Das Produkt stellt keine Funktionen zur Verfügung, die eine Anpassung an verschiedene Benutzer ermöglichen. Auch der Quellcode des Programms wird nicht mitgeliefert.

1	2	3	4	5	6
-	-	-	-	-	-

5.4 Visual Paradigm for UML 6.4

5.4.1. Methodenbasierte Kriterien

a) Modellierung der Objektstruktur – Objektstatik

Neue Objekte werden sowohl im Zeichenbereich als auch im Diagram Navigator angelegt. Beim Anlegen selbst werden Namensgleichheiten vermieden (zB. Klasse und Klasse2). Vergibt man den Namen neu, so sind auch Namensgleichheiten möglich. Alle Informationen sind im Diagrammfenster in Form eines Baummenüs (hier Diagram Navigator) dargestellt. Dort kann man auch die Eigenschaften der Komponenten verändern. Im Zeichenbereich selbst können von Attributen Name, Sichtbarkeit, Typ und der Startwert angegeben werden. Darüberhinaus können noch weitere Eigenschaften wie Constraints, Comments etc. angepasst werden. Für Methoden ist die Definition von Constraints, Parametern, Pre- und Post-Conditions und einiger weiterer Eigenschaften möglich.

1	2	3	4	5
+	+	+	+	+

b) Modellierung der Systemstruktur – Systemstatik

Rekursive Beziehungen können mit Visual Paradigm modelliert werden. Einmal erfasste Beziehungen können problemlos mittels drag-and-drop verändert werden. Object Constraint Language (OCL) – Annotationen werden von diesem Tool

unterstützt. Die Einteilung von Klassen in Kategorien unabhängig von Packages ist nicht möglich.

1	2	3	4
+	+	-	+

c) Dynamisches Objektverhalten – Objektdynamik

Es können für jede Klasse Zustandsdiagramme erstellt werden, die umkehrbar eindeutig mit der jeweiligen Klasse verbunden sind. Ebenso bietet Visual Paradigm die Möglichkeit, Bedingungen für Zustandsübergänge zu definieren. Aktivitäten und Ereignisse können im Sinne einer genaueren Beschreibung verfeinert werden. Sequenzdiagramme nehmen Bezug auf Methoden bzw. Attribute des statischen Objektmodells.

1	2	3	4	5	6	7
+	+	+	+	o	-	+

d) Zusammenhang zwischen den Modellen

Jedes Zustandsdiagramm ist umkehrbar eindeutig mit einem Objekt bzw. einer Klasse verbunden. Im Browser werden Zustandsdiagramme nicht unter der jeweiligen Klasse angezeigt, sondern im Unterordner „State Machine Diagram“. Das dynamische Verhalten einer Klasse wird nicht weitervererbt. Man kann sich bei der Zustandsmodellierung auf Attribute und Klassen des Zustandsdiagramms beziehen.

1	2	3	4
+	-	+	+

e) Abstraktionsebenen / Sichten

Bei Visual Paradigm können die Sichtbarkeit, Attribute, Anfangswerte, Methoden, Methodensignaturen ein- bzw. ausgeblendet werden. Die Einstellungen können über das Kontextmenü unter Präsentationsoptionen vorgenommen werden. Mehoden und Attribute können auch über ein „-„, das bei Berührung der Klasse mit der Maus erscheint ausgeblendet werden. Objektkomponenten können für eine Variation der Genauigkeit der Darstellung verändert werden.

1	2	3	4
+	+	+	-

f) Vorgangsreihenfolge bei der Entwicklung

Die Verwaltung des Repositorys ist unabhängig von den Diagrammsichten. Er muss sich daher nicht an bestimmte Vorgehensweisen halten sich der Benutzer nicht an bestimmte Vorgehensmodelle halten. Der Benutzer kann die Informationen zu existierenden Komponenten auch direkt im Diagramm-Navigator erfassen.

Um neue Elemente zu generieren, muss der Benutzer auf „Modell“ (Model) oder auf „Klassenverzeichnis“ (Class Repository) zugreifen.

1	2	3
+	o	+

g) Unterstützung bei der Methodenanwendung

Das Tool besitzt eine mitgelieferte Hilfe im HTML-Format. Informationen zu UML selbst kann man daraus nicht entnehmen. Auf der Herstellerseite findet man aber genügend Lernhilfen und Informationen zu UML. Einige Notationselemente unterscheiden sich geringfügig von UML-Standard. Eine Konsistenz- und Integritätsprüfung kann nicht explizit aufgerufen werden. Das Programm achtet jedoch selbständig darauf. Legt man zum Beispiel eine Klasse, die schon existiert, noch einmal an, so werden ihr automatisch alle Eigenschaften, Attribute und Methoden der existierenden Klasse zugeteilt. Manipulationen an einer Klasse werden automatisch für alle Klassen, die den gleichen Namen tragen, übernommen.

1	2	3
-	o	o

5.4.2. Funktionale Kriterien

a) Datenhaltung

Das Speichern der Modellierungsinformationen erfolgt im *Class Repository*. In Form eines Baummenüs kann auf die Daten zugegriffen werden. Werden Informationen aus den Diagrammen entfernt, löschen sich auch dieselben Informationen im Modell Explorer automatisch. Das Manipulieren von Daten ist auch direkt im Repository möglich. Die Daten werden im vpp-Format gespeichert. Es ist jedoch zusätzlich ein Export in XML und XMI sowie als Image möglich.

Importiert werden können außer Visual Paradigm-Projekten Rose Projekte, XML, XMI, Erwin Projekte (XML) und MS Word to Use Case Model.

1	2	3	4	5
+	+	+	+	+

b) Browsing-Eigenschaften

Eine Suchfunktion existiert. Sie kann auch mit dem Windows-üblichen Shortcut „Strg+f“ aufgerufen werden. Es besteht die Möglichkeit, die Suche auf aktive, geöffnete oder das aktuelle Diagramm einzuschränken oder alle Diagramme zu durchsuchen. Suchergebnisse werden in Form einer Baumstruktur im Message-Fenster am unteren Bildschirmrand ausgegeben. Um direkt zu einem Element zu gelangen, kann der Befehl „Springe zu Element“ verwendet werden. Patternmatching wird nicht unterstützt. Verwendet man den Befehl „Springe zu Element“, so

werden alle möglichen Elemente vorgeschlagen, aus denen man wählen kann. Bei der Suche können nur wenige Einschränkungen vorgenommen werden. Eine Definition eigener Filter ist nicht möglich. Durch Doppelklick auf ein Suchergebnis gelangt man zu den Eigenschaften des Elements. Weitere Informationen (Beziehungen, Ober-/Unterklassen…) werden nicht angezeigt. Änderungen von Informationen sind auch direkt im Repository möglich. Diese werden automatisch in die Diagramme übernommen. Klassenhierarchien werden in textueller Form als Baumstruktur im Diagramm-Navigator dargestellt.

1	2	3	4	5	6	7	8	9	10	11
+	-	+	o	o	-	+	+	+	-	+

c) Navigation im Systemmodell

Eine rasche Navigation zwischen den Diagrammen ist über den Diagramm-Navigator problemlos möglich. Über ein Baummenü sind alle Diagramme erreichbar. Will man in ein Subdiagramm einer Klasse wechseln, so muss man sie mit der Maus berühren und das Symbol „Unterdiagramme“ anklicken. Man gelangt zu einer Auswahl der zur Verfügung stehenden Diagramme. Umgekehrt kann man ein Überdiagramm über das Kontextmenü und „Open Parent Model“ erreichen.

1	2	3
+	+	o

d) Konsistenzprüfung / Integrität

Assoziationen oder Übergänge ins Leere sind nicht möglich. Es ist nicht möglich rekursive Vererbungen zu erstellen. Es können Modellelemente mit gleicher Bezeichnung angelegt werden. Jedoch achtet das System darauf, dass alle Elemente des gleichen Typs, die den gleichen Namen tragen, gleich sind. Manipulationen an einer Klasse werden automatisch auf die anderen Klassen mit gleichem Namen übertragen. Die referentielle Integrität im System ist vorhanden. Werden Teile aus einem Diagramm gelöscht, werden sie nicht automatisch aus der Datenbasis gelöscht. Wird ein Element aus einem Diagramm gelöscht, das nirgendwo anders mehr benötigt wird, fragt, das Tool nach, ob das Element auch aus dem Repository entfernt werden soll.

1	2	3	4	5	6
+	+	-	-	-	+

e) Report-Ausgabe

Visual Paradigm for UML erlaubt die Ausgabe von Reports in den Formaten HTML, PDF und als Word-Dokument.

1	2	3	4	5	6	7	8
+	+	+	+	-	-	-	+

f) Datenbankschema-Ausgabe

Über das Kontextmenü und „Senden an“ besteht die Möglichkeit, Daten an ein ORM-Diagramm weiterzuleiten. Von dort hat man die Möglichkeit, das ORM-Diagramm weiter auszubauen. Ab diesem Punkt kann man auch mit Entitäten und Beziehungen arbeiten. Über die Eigenschaften der Entitäten können auch Schlüsselattribute gesetzt werden. Generiert man aus einer Klasse eine Entität, wird leider nur der Name übernommen. Attribute oder Beziehungen werden nicht berücksichtigt. Für die Arbeit mit Datenbankschemata ist man allerdings mit der Professional Edition oder der Enterprise Edition besser bedient.

1	2	3	4	5
o	o	+	-	o

g) Diagramm-Ausgabe

Die Ausgabe von Diagrammen ist in Visual Paradigm for UML ist sowohl an den Drucker als auch als Image, XML oder XMI möglich. Die Ausgabe an den Drucker bietet die Optionen Hoch- und Querformat sowie die Auswahl des Papierformats. Weitere einstellbare Optionen sind nicht vorhanden. Die Ausgabe entspricht der Bildschirmansicht.

1	2	3	4	5
+	+	+	-	+

h) Offenheit – Import / Export

Visual Paradigm for UML ist Teil der Visual Paradigm Suite.

Weitere Module der Visual Paradigm Suite sind:

- Smart Development Environment (SDE)
- DB Visual Architect (DB-VA)
- Business Process Visual Architect (BP-VA)
- Agilian (AG)
- DB Visual Architect SQL (DB-VA SQL)

Für diese Module benötigt man jedoch jeweils eigene Lizenzen. Man kann auch die toolfremden Formate XML, XMI, RoseProject und ERwinProject (XML) einlesen und bearbeiten. Durch Ausgabe in toolfremde Formate wie XML können Ausgaben von Visual Paradigm for UML auch von Produkten anderer Hersteller, die XML interpretieren können, gelesen werden.

1	2	3	4	5
+	+	+	-	+

i) Codegenerierung

Die automatische Codegenerierung wird nur für Java, C#, VB.NET, PHP 5.0, IDL, C++, Delphi, Perl, XML, Python, Objective-C ADA und Ruby unterstützt. Für jede im Modell vorkommende Klasse wird eine entsprechende Codierung vorgenommen. Attribute und Typen werden übernommen. Beziehungen werden durch Attribute im Sourcecode realisiert. Methodengerüste und eventuelle Zugriffsbeschränkungen werden automatisch erstellt. Der Ausmaß der Codegenerierung lässt sich nicht bestimmen, Parameter z. B. Prefixe für Attribute können jedoch gesetzt werden. Händisch in den Source-Dateien hinzugefügter Code bleibt bei erneuter Codegenerierung erhalten, wird jedoch nicht explizit gekennzeichnet.

1	2	3	4	5	6	7	8	9	10	11
+	+	+	+	+	+	+	+	o	+	-

j) Reverse-Engineering

Reverse Engineering ist für Java, C++, CORBA IDL, .NET, XML, ADA, JDBC, Hibernate, Python, PHP und Objective-C möglich. Die Diagramme wurden ohne Probleme erstellt. Vergibt man keinen Package-Namen, so wird automatisch ein Default-Package angelegt. Auch Code, der nicht von Visual Paradigm for UML erstellt wurde, kann eingelesen werden. Auch Code innerhalb von Methoden kann eingelesen werden und in einem internen Editor bearbeitet werden.

1	2	3
+	+	+

k) Test / Verifikation

Eine Möglichkeit zum Testen bzw. zur Simulation auf Modellebene ist in Visual Paradigm for UML nicht vorgesehen.

1	2
-	-

l) Maßzahlen und Statistiken

Die Generierung von Maßzahlen und Statistiken ist mit Visual Paradigm for UML nicht möglich. Es können keine Aussagen über Anzahl der Klassen, Attributen, Methoden, usw. gemacht werden.

1	2	3
-	-	-

m) Multi-User – Eigenschaften

Das Tool selbst kann nicht von mehreren Usern gleichzeitig benutzt werden. Es ist aber möglich, Daten auf einem gemeinsamen Server abzulegen, auf den mehrere zugreifen können. Für die Arbeit mit dem Server werden einige Funktionen wie Commit, Update angeboten. Projekte können vom Server geholt, lokal bearbeitet und auf den Server zurückgespielt werden. Der Server übernimmt dabei auch das Konfliktmanagement. Das Locking von ganzen Diagrammen ist möglich.

1	2	3	3
o	-	+	+

n) Security

Der Benutzer braucht sich nach dem Programmstart nicht anmelden. Am Server ist jedoch eine Anmeldung notwendig. Dort können auch Zugriffsrechte individuell an die User vergeben werden. Bei Bedienungsfehlern werden dem Benutzer entsprechende Meldungen geliefert und das UML-Werkzeug bleibt in einen definierten Zustand. Nach dem Löschen von Modellierungsinformationen muss der Benutzer diese Aktion noch einmal bestätigen. Dadurch soll ein unbeabsichtigtes Löschen von Daten verhindert werden. Durch die Undo-Funktion kann auch ein Löschen von Informationen ganz einfach rückgängig gemacht werden.

1	2	3	4	5	6	7	8	9
o	+	-	-	-	+	+	+	+

o) Multi-Projekt – Eigenschaften / Versionsmanagement

Eine zeitlich parallele Abwicklung von mehreren Projekten ist nicht möglich. Es kann nur ein Projekt zu einem Zeitpunkt geöffnet sein, außer man startet die Applikation öfters. Am Server können natürlich mehrere Aufgaben am Server liegen. Ein Austausch von Daten kann über den Server durchgeführt werden. Ein Versionskontrollsystem ist am Server integriert.

1	2	3	4	5
o	o	-	+	-

5.4.3. Ergonomie

a) Benutzeroberfläche

Das Aussehen von Visual Paradigm for UML ist bezüglich der Fensteraufteilung anderen Entwicklungsprogrammen ähnlich. Das GUI ist trotz der vielen Funktionen übersichtlich gestaltet. Die Dialoge und Eingabemasken sind übersichtlich und einfach und intuitiv zu bedienen. Die Namensgebung der Menüs und Funktionen ist klar definiert und ist selbstbeschreibend, einheitlich und widerspruchsfrei. Deaktivierte Funktionen sind grau hinterlegt. Bei länger dauernden Aktionen bekommt der Benutzer vom System ein Feedback in Form eines Fortschrittsbal-

kens. Um genügend Platz für die Erstellung von Diagrammen am Bildschirm zu haben, können Teilfenster zugeklappt oder geschlossen werden.

1	2	3	4	5	6	7
+	+	+	+	+	+	+

b) Tool-Dokumentation

Bei der Installation von Visual Paradigm for UML besteht die Möglichkeit, Dokumentationen im PDF- und HTML-Format herunterzuladen, die auch vom Programm aus aufrufbar sind. Die Dokumentationen sind in englischer Sprache verfasst. Mit Hilfe von Screenshots werden die einzelnen Funktionen ausführlich beschrieben und dargestellt. Dadurch wird das Erlernen des Umgangs mit dem Tool erleichtert. Eine schriftliche Installationsanleitung kann separat heruntergeladen werden. Die Dokumentation beginnt mit dem Start des Programms und behandelt jedes einzelne Fenster. Man kann daher von einer Anleitung für unerfahrene Benutzer sprechen. Die Dokumentation kann auch noch nach der Installation (oder auch ohne Installation) von der Hersteller-Homepage heruntergeladen werden.

1	2	3	4	5	6	7	8	9	10	11	12
+	+	-	-	+	+	-	+	+	+	-	-

c) Usability

Das Tool ermöglicht ein rasches und effizientes Arbeiten. Die übersichtlich angeordneten Werkzeuge und die vorhandenen Shortcuts lassen den Anwender schnell auf die von ihm gewünschten Funktionen zugreifen. Fehlermeldungen sind vorhanden und verständlich. Shortcuts sind vorhanden. Das Belegen von Funktionstasten ist leider nicht möglich. Die Kontextmenüs sind übersichtlich und bieten die wichtigsten Funktionen an. Ungültige Eingaben und Operationen werden mittels Fehlermeldung behandelt. Die vorhandenen Symbolleisten sind übersichtlich angeordnet. Man hat auch für jeden Diagrammtyp ein Symbol zur Verfügung. Das vereinfacht die Erstellung neuer Diagramme vor allem im Gegensatz zu Objecteering. Der Fortschritt längerer Systemoperationen (wie z.B. Laden oder Speichern von Projekten oder das Generieren von Code) wird mittels Fortschrittsbalken angezeigt. Eine Zoomfunktion für Diagramme ist ebenfalls vorhanden. Mehrfenstertechnik wird unterstützt. Die Betrachtung mehrerer Diagramme zur gleichen Zeit ist daher möglich. Eine Mehrfachauswahl von Elementen ist mit gedrückter Shift-Taste möglich. Visual Paradigm for UML ist in den Sprachen Deutsch, Englisch, Japanisch und Chinesisch verfügbar.

1	2	3	4	5	6	7	8	9	10	11
-	o	+	-	+	+	+	-	+	+	+

d) Hilfefunktionen

Die Hilfefunktion ist nur über das Menü erreichbar und kann nicht über die Funktionstaste F1 aufgerufen werden. Eine hypertextartige Hilfe ist vorhanden und in einer Baumstruktur nach Themengebieten gegliedert. Zusätzlich ist es möglich, nach Themengebieten zu suchen. Eine Hot-Help für die verschiedenen Symbole

nicht verfügbar. Beim Berühren eines Symbols wird nur dessen Name und Symboltyp angezeigt.

1	2	3	4
-	+	+	o

e) Graphikeditor / Darstellung

Eine Anpassung der Symbole ist bei Visual Paradigm for UML nicht möglich. Der Benutzer kann jedoch über den Shape-Editor eigene Elemente zeichnen. Die am Bildschirm dargestellten Diagramme entsprechen dem späteren Ausdruck. Die Farben der verschiedenen Notationselementen sind für jedes Element frei wählbar. Elemente können horizontal bzw. vertikal automatisch ausgerichtet werden. Dabei achtet das Tool auch darauf, dass sich Linien nicht überschneiden. Auch eine Zoomfunktion ist vorhanden. Symbole und Texte können nicht rotiert werden. Die Größe eines Elements ist nicht von dessen Inhalt abhängig, sondern einheitlich, solange man sie nicht verändert. Zeilenumbrüche in der Beschriftung von Elementen können durch Strg+Return herbeigeführt werden. Beim Speichern und erneutem Laden von Diagrammen wird das Layout beibehalten. Die Schriftgröße kann verändert werden und ist gut lesbar. Außerdem sind Schriftart, Schriftstil und Farbe wählbar. Assoziationen können durch nachträglich festgelegte Eckpunkte verändert werden. Strichstärke und Linienart können über das Kontextmenü verändert werden. Eine Refresh-Funktion ist nicht vorhanden.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-	+	+	+	+	+	-	+	+	+	+	+	+	+	-

f) Graphikeditor / Bedienung

Das Einfügen neuer Elemente in den Zeichenbereich erfolgt bei Visual Paradigm for UML durch Klick auf das Symbol in der Symbolleiste und anschließenden Klick in den Zeichenbereich. Hierfür werden die zur Verfügung stehenden Elemente und Werkzeuge am linken Rand des Zeichenbereichs eingeblendet. Nach dem Anlegen der Notationselemente können diese sofort textuell beschrieben werden. Eine Funktion zum vielfachen Einfügen der desselben Elemente hintereinander ist leider nicht vorhanden. Die Bedienung erfolgt intuitiv. Das gilt auch für die Manipulation der Diagrammelemente, wie z.B. das Verschieben der Elemente. Objekte können automatisch horizontal bzw. vertikal an Gitterpunkte ausgerichtet werden. Die Abstände der Gitternetzlinien sind sowohl in der Höhe als auch in der Breite frei wählbar. Um mehrere Symbole zu bearbeiten, können diese mit der Maus umrahmt werden. Auch eine Gruppierung ist möglich. Mittels Undo- und Redo- Funktionen können Bearbeitungsschritte schnell und komfortabel rückgängig bzw. wiederholt werden. Die Funktionen der Zwischenablage werden ebenfalls unterstützt (Copy/Cut und Paste). Durch die Entf-Taste können Symbole aus den Diagrammen wieder entfernt werden.

1	2	3	4	5	6	7	8	9	10	11	12	13	14
+	+	-	+	+	+	+	+	+	+	+	+	+	+

g) Performance

Das Programm ist im Gegensatz zu anderen Programmen etwas langsamer, unterstützt dafür aber auch mehr Funktionen. Auch das Arbeiten mit großen Modellen wird nach einiger Zeit ein wenig zeitversetzt. Im Großen und Ganzen ist die Performance aber zufriedenstellend. Auch die Generierung von Code erfolgt nur mit mittelmäßiger Geschwindigkeit.

1	2	3	4
o	o	+	o

h) Adaptivität

Im Bereich der Adaptivität ist auch Visual Paradigm for UML mangelhaft. Das Produkt stellt keine Funktionen zur Verfügung, die eine Anpassung an den Lernfortschritt des Users ermöglichen. Ebenso wenig verändern oder anpassen lassen sich Symbolleisten und Menüs. Auch der Quellcode des Programms wird nicht mitgeliefert.

1	2	3	4	5	6
-	-	-	-	-	-

5.5 Magic Draw

5.5.1. Methodenbasierte Kriterien

a) Modellierung der Objektstruktur – Objektstatik

Die neuen Objekten und Informationen werden automatisiert im Projektverzeichnis gespeichert. Beim Anlegen von neuen Klassen erfolgt keine Überprüfung auf Namensgleichheit. Es ist jedoch nicht möglich Elemente von Klassen (Attribute, Methoden) gleich zu benennen. Neue Objekte und Information werden im Containment Baum in einer Baumstruktur dargestellt. Bearbeit werden können die Elemente sowohl im Zeichenbereich als auch im Containment Baum. Für jedes Attribut eines Klassendiagramms kann der Name und Typ definiert werden. Für Methoden können darüber hinaus noch der Rückgabetyp, Parameter sowie Vor- und Nachbedingungen festgelegt werden.

1	2	3	4	5
+	+	+	+	+

b) Modellierung der Systemstruktur – Systemstatik

Magic Draw ermöglicht die Darstellung von rekursiven Beziehungen. Assoziationen können nach ihrer Erstellung jederzeit nachträglich geändert bzw. modifiziert werden. Diese nachträgliche Änderung kann mittels drap-and-drop, oder über die Objekteigenschaften erfolgen. Object Constraint Language (OCL) – Annotationen

werden von diesem Tool unterstützt. Die Einteilung von Klassen erfolgt mittels Packages. Klassen können somit kategorisiert werden.

1	2	3	4
+	+	+	+

c) Dynamisches Objektverhalten – Objektdynamik

Für jede Klasse kann in Magic Draw ein Zustandsdiagramm erstellt werden, das umkehrbar eindeutig mit der jeweiligen Klasse verbunden ist. Ebenso besteht die Möglichkeit Bedingungen für Zustandsübergänge zu definieren. Alle Aktivitäten und Ereignisse können im Sinne einer genaueren Beschreibung verfeinert werden. Diese Beschreibung kann über die jeweiligen Objekteigenschaften festgelegt werden. Das Tool ermöglicht auch die Darstellung von geschachtelten Zustandsdiagrammen. Sequenzdiagramme nehmen Bezug auf Methoden bzw. Attribute des statischen Objektmodells.

1	2	3	4	5	6	7
+	+	+	+	+	+	+

d) Zusammenhang zwischen den Modellen

Jedes Zustandsdiagramm kann in Magic Draw umkehrbar eindeutig mit einem Objekt bzw. einer Klasse verbunden werden. Im Containment Baum werden Zustandsdiagramme unter der jeweiligen Klasse angezeigt. Dabei werden Zustandsdiagramme nur unter der verbundenen Klasse und nicht unter Subklassen angezeigt.

1	2	3	4
+	+	-	-

e) Abstraktionsebenen / Sichten

In Magic Draw lässt sich der Detaillierungsgrad Klassen individuell anpassen. Attribute und Methoden sowie deren Typen, Rückgabetypen, Parameter, Parametertypen etc. können je nach Anforderung ein- bzw. ausgeblendet werden. Dabei kann der Benutzer zwischen einer Standard- und einer Expertenansicht wählen. Die Diagrammanzeige kann für jede Klasse separat, oder auch generell für alle Klassen angepasst werden. Der Benutzer kann hierbei den Detaillierungsgrad selbst bestimmen.

1	2	3	4
+	+	+	+

f) Vorgangsreihenfolge bei der Entwicklung

Das Tool zwingt den Benutzer bei der Erstellung von Klassen und Informationen sich an eine bestimmte Reihenfolge zu halten. Das Spektrum der zu erzeugenden Elemente orientiert sich an der Modellierungsebene. Zuerst muss ein Paket erstellt werden, in diesem Paket können Klassen erzeugt werden und anschließend

Attribute und Methoden. Objekte können zuerst im Containment Baum angelegt werden und anschießend im Zeichenbereich übernommen werden. Werden Elemente im Zeichenbereich erstellt, erfolgt die Aufnahme im Containment Baum automatisch.

1	2	3
+	+	+

g) Unterstützung bei der Methodenanwendung

Das Tool besitzt eine mitgelieferte UML-Hilfe. Zusätzlich findet man auf der Herstellerseite Lernhilfen und Informationen zu UML. Unter der Rubrik Hilfe findet man weiters einen Link zu www.uml.org. Einige Notationselemente unterscheiden sich geringfügig vom UML-Standard. Eine Konsistenz- und Integritätsprüfung ist teilweise vorhanden.

1	2	3
o	+	o

5.5.2. Funktionale Kriterien

a) Datenhaltung

Das Speichern der Modellierungsinformationen erfolgt im *Containment Baum*. In Form eines Baummenüs kann auf die Daten zugegriffen werden. Werden Informationen aus dem Containment Baum entfernt, werden die entsprechenden Objekte auch aus dem Zeichenbereich gelöscht. Das Manipulieren von Daten ist auch direkt im Containment Baum möglich. Die Projekte können als MDZIP MDXML, XML, oder XMLZIP gespeichert werden. Somit können UML-Werkzeuge mit XML Unterstützung mit diesen Daten arbeiten. Auch der Import kann nur über die angeführten Datentypen erfolgen.

1	2	3	4	5
+	+	+	+	+

b) Browsing-Eigenschaften

In Magic Draw existieren zwei Suchfunktionen, eine Schnellsuche und eine normale Suche. Mit beiden Suchfunktionen kann sowohl nach Elementen, als auch nach Diagrammen gesucht werden kann. Ein Pattern Matching ist für die schnelle Suche integriert. Es kann auch nach Elementen per Anfangsbuchstabe gesucht werden. Die Ergebnisliste berechnet sich dynamisch. Für die Suche können Einschränkungen und Gültigkeitsbereiche definiert werden.

Die Ergebnisse der Suche werden in einem Suchbaum angezeigt. Dieser entspricht dem Aufbau des Containment Baumes und zeigt die Resultate geordnet in diversen Kategorien. Die Ergebnisse können direkt in dem Ergebnissuchbaum modifiziert werden.

1	2	3	4	5	6	7	8	9	10	11
+	+	+	+	+	+	+	+	+	+	+

c) Navigation im Systemmodell

In Magic Draw kann man über den Containment Baum zwischen den einzelnen Diagrammen navigieren. Zusätzlich kann die Navigation über Tabs des Diagrammeditors erfolgen. Im Containment Baum sind alle Diagramme über ein Baummenü erreichbar. Es ist ebenfalls möglich, über ein Zustandsdiagramm zum entsprechenden Klassendiagramm zu wechseln. Informationen zu den verschiedenen Elementen können durch Doppelklick auf das entsprechende Element aufgerufen bzw. modifiziert werden.

1	2	3
+	+	+

d) Konsistenzprüfung / Integrität

Assoziationen ins Leere sind nicht möglich. Sobald man eine Assoziation, die ins Leere gehen soll anlegt, wird automatisch eine neue Klasse ans Ende der neuen Assoziation angelegt. Es ist nicht möglich, rekursive Vererbungen zu erstellen. In einem Paket kann man mehrere Klassen mit demselben Namen anlegen. In einer Klasse werden hingegen die Bezeichnungen der einzelnen Elemente auf Namensgleichheit überprüft.

Das System prüft nicht auf Ähnlichkeiten zwischen Bezeichnungen. Eine Konsistenzprüfung kann automatisch durchgeführt werden. Nach dem Löschen eines Elementes in der Zeichenebene bleibt das Element weiter in Containment Baum verfügbar.

1	2	3	4	5	6
+	+	-	-	+	+

e) Report-Ausgabe

In Magic Draw können Reports automatisiert erstellt werden. Das Tool unterstützt die Erzeugung von Reports in den Formaten HTML, PDF und RTF. Die Erstellung der Reports erfolgt mit dem Report Assistenten.

1	2	3	4	5	6	7	8
+	+	+	+	-	-	+	+

f) Datenbankschema-Ausgabe

Mit dem Modell Transformation-Assistenten kann man ein Datenbankschema für relationale und objektorientierte Datenbanken auf Basis der Informationen des Objektmodells generieren. In den Attributeigenschaften kann festgelegt werden, ob es sich um ein eindeutiges Attribut für die Klasse handelt. Weiters können für Attribute Wertebereiche definiert werden. Alle Informationen des Klassendiagramms werden im Datenbankschema wiedergeben.

1	2	3	4	5
+	+	+	+	+

g) Diagramm-Ausgabe

Die Diagrammausgabe von Magic Draw orientiert sich an den Möglichkeiten von Microsoft Office Produkten. Neben der Einstellung der gewünschten Druckgröße und der Diagrammausrichtung kann jedoch kein Abstraktionsniveau eingestellt werden. Die Ausgabe entspricht der Bildschirmansicht.

1	2	3	4	5
+	+	+	-	+

h) Offenheit – Import / Export

Magic Draw kann mit einigen weiteren Plug-Ins erweitert werden. Diese können gegen ein Entgelt vom Hersteller bezogen werden. Es ist nicht möglich, Grafik- und Textformate in den Zeicheneditor einzulesen. Sofern Tools von Drittherstellern in der Lage sind Formate des Exports wie z.B. XML zu lesen, können sie mit den Daten von Magic Draw arbeiten.

1	2	3	4	5
+	-	-	-	+

i) Codegenerierung

Die automatische Codegenerierung wird nur für Java, EJB, C#, C++, CORBA IDL, DDL, WSDL und XML Schema unterstützt. Für jede im Modell vorkommende Klasse wird eine entsprechende Codierung vorgenommen. Attribute und Typen werden übernommen. Methodengerüste und eventuelle Zugriffsbeschränkungen werden automatisch erstellt. Das Ausmaß der Codegenerierung lässt sich nicht bestimmen.

1	2	3	4	5	6	7	8	9	10	11
+	+	+	+	+	+	+	+	-	-	-

j) Reverse-Engineering

Reverse Engineering ist für Java, C#, C++, CORBA IDL, EJB 2.0, DDL, CIL (MSIL), WSDL, und XML Schema möglich. Diagramme werden aus Sourcecode-Dateien ohne Probleme erstellt. Es werden automatisch Pakete angelegt. Auch Code, der nicht von Magic Draw erstellt wurde, kann eingelesen werden. Eben-

falls kann Code innerhalb von Methoden eingelesen werden und in einem internen Editor bearbeitet werden.

1	2	3
+	+	+

k) Test / Verifikation

Eine Möglichkeit zum Testen bzw. zur Simulation auf Modellebene wird von Magic Draw nicht unterstützt.

1	2
-	-

l) Maßzahlen und Statistiken

Magic Draw unterstützt die Generierung von System-, UML- und Anforderungsmetriken. Es können Aussagen über Anzahl der Klassen, Attributen, Methoden, usw. gemacht werden. Weiters können Optionen für die einzelnen Metriken definiert werden.

1	2	3
+	+	+

m) Multi-User – Eigenschaften

Mit der Verwendung des Magic Draw Teamwork Servers können mehrere Entwickler simultan an einem Modell arbeiten. Der Einsatz dieses Servers ermöglicht weiters, ein Konfigurationsmanagement, den kontrollierten Zugriff auf Objekte und Elemente sowie den Remote Zugriff auf die Modelle.

1	2	3	3
+	+	+	+

n) Security

Das Tool als solches unterstützt keine Benutzerverwaltung. Der Benutzer braucht sich nach dem Programmstart nicht anzumelden. Eine solche Sicherheitsfunktion ist nicht implementiert. Auch das Protokollieren von Änderungen durch den Benutzer ist nicht umgesetzt. Das Tool verfügt über eine integrierte Datensicherung und eine Recoveryfunktion. Es ist möglich, das automatische Intervall für die Systemspeicherung anzupassen. Bei Falscheingaben wird die Systemoberfläche eingefroren und der Benutzer erhält eine Fehlermeldung.

Das Löschen von Modellierungsinformationen muss vom Benutzer ein zweites Mal bestätigt werden.

Für den Fall, dass es trotzdem zu einem unbeabsichtigten Löschen kommt, verfügt das Tool über eine Rückgängig-Funktion.

1	2	3	4	5	6	7	8	9
-	-	-	+	+	+	+	+	+

o) Multi-Projekt – Eigenschaften / Versionsmanagement

Es ist nicht möglich gleichzeitig verschiedene Projekte zu bearbeiten, da zu einem Zeitpunkt immer nur ein Projekt geöffnet sein kann. Der Austausch von Modellierungsinformationen zwischen verschiedenen Projekten kann über eine Erstellung aus einem bestehenden Projekt erfolgen. Eine Versionierung wird bei der Standalone Version des Tools nicht unterstützt.

1	2	3	4	5
-	+	+	-	-

5.5.3. Ergonomie

a) Benutzeroberfläche

Das Erscheinungsbild von Magic Draw orientiert sich standardmäßig am Microsoft Office Stil und ermöglicht dem Benutzer eine intuitive Handhabung des Tools. Die Bezeichnungen der verschieden Menüpunkte sind selbst beschreibend und in Deutsch. Der Benutzer kann in den Optionen unter der Rubrik Look and Feel das Layout selbst bestimmen. Dies ermöglicht auch nicht Microsoft Benutzern ein intuitives Arbeiten. Die Dialoge und Eingabemasken sind leicht zu bedienen und gut beschrieben. Bei Aktionen die eine längere Verarbeitungszeit benötigen wird ein entsprechender Fortschrittsbalken angezeigt. Menüpunkte die zurzeit nicht zur Verfügung stehen werden ausgegraut. Auch der Aufbau der Arbeitsoberfläche kann selbst bestimmt werden. Der Benutzer kann eine Auswahl der anzuzeigenden Fenster festlegen und deren Anordnung bestimmen.

1	2	3	4	5	6	7
+	+	+	+	+	+	+

b) Tool-Dokumentation

Mit dem Herunterladen von Magic Draw wird auf dem Rechner auch eine Hilfe installiert, die vom Programm aus geöffnet werden kann. Das Benutzerhandbuch ist in englischer Sprache verfasst und als PDF-Dokument verfügbar. Weiters werden im Tool Links zu Onlinedokumentationen, Demos, der Homepage des Herstellers, aber auch zu allgemeinen Themen wie UML angeboten. Mit Hilfe von Screenshots werden die einzelnen Funktionen ausführlich beschrieben und dargestellt. Die Hilfe und das Benutzerhandbuch verfügen über ein strukturiertes Inhaltsverzeichnis. Außerdem gibt es in diesen Dokumenten auch eine schriftliche Anlei-

tung zur Installation, sowie eine „Erste-Schritte“ Anleitung für unerfahrene Benutzer. Weitere Hilfestellungen findet der Benutzer auf der Herstellerhomepage in Form von PDF-Dokumenten, die man auf seinen Rechner herunterladen kann.

1	2	3	4	5	6	7	8	9	10	11	12
+	+	-	+	+	+	-	+	+	+	+	+

c) Usability

Das Tool ermöglicht ein rasches und effizientes Arbeiten. Übersichtlich angeordnete Werkzeuge und die vorhandenen Shortcuts lassen den Anwender schnell auf die von ihm gewünschten Funktionen zugreifen. Die Systemausgaben in Form von Fehlermeldungen sind klar und verständlich. Das Belegen von Funktionstasten ist leider nicht möglich. Die Kontextmenüs sind übersichtlich und funktionell.

Fehlerhafte Eingaben werden vom System abgefangen und mit dem entsprechenden Feedback ausgegeben. Die vorhandenen Symbolleisten sind übersichtlich angeordnet. Der Benutzer kann die Anordnung der Symbolleisten individuell an seine Bedürfnisse anpassen. Längere Systemoperationen (wie z.B. Laden oder Speichern von Projekten) werden mittels Fortschrittsanzeigen und weiteren Informationen angezeigt.

In Magic Draw ist es möglich mehrere Fenster gleichzeitig zu betrachten. Die einzelnen Fenster verfügen über eine Zoomfunktion. Mit Drücken der Shift-Taste können mehrere Elemente gleichzeitig ausgewählt werden. Das Tool ist in englischer und deutscher Sprache verfügbar.

1	2	3	4	5	6	7	8	9	10	11
+	+	+	-	+	+	+	+	+	+	+

d) Hilfefunktionen

Das Tool bietet neben dem Eintrag im Menü eine kontextintensive Hilfe, die über die Taste F1 erreichbar ist. Das Werkzeug verfügt neben dem Handbuch über eine Hypertextsuche. Ebenso ist ein Suchen nach bestimmten Themen möglich. Die Funktionalität Hot-Help für die verschiedenen Symbole ist verfügbar.

1	2	3	4
+	+	+	-

e) Graphikeditor / Darstellung

Bei Magic Draw können mit Rechtsklick auf ein Symbol die Symboleigenschaften angepasst werden. Es gibt keine Zeichenelement wie Rechtecke oder Ovale, neben den diagrammabhängigen Darstellungsmöglichkeiten. Die am Bildschirm dargestellten Diagramme entsprechen dem späteren Ausdruck. Die Farben der verschiedenen Notationselemente können für jedes Element einzeln verändert werden. Elemente können horizontal bzw. vertikal automatisch ausgerichtet werden. Magic Draw besitzt ein stufenloses Zoom, das im Diagrameditor mit dem

Lupe-Symbol aufgerufen werden kann. Es ist möglich Texte zu rotieren und ihren Schriftgrad zu bearbeiten.

Die Größe und Anordnung von Symbolen kann nach ihrer Erstellung verändert werden. Auch die Schriftgröße kann verändert werden. Assoziationen können durch nachträglich festgelegte Eckpunkte verändert werden. Die Strichstärke und Linienart kann man nicht verändern. Ein Refresh ist nicht vorhanden.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
+	-	+	+	+	+	+	+	+	+	+	+	+	+	-

f) Graphikeditor / Bedienung

Bei Magic Draw können Diagramme durch die Verwendung des Containment Baumes, oder direkt durch die Verwendung der Werkzeugliste erstellt werden. Verwendet man die Werkzeugliste können die einzelnen Elemente direkt in den Zeichenbereich gezogen werden. Nach der Platzierung an der gewünschten Position kann sofort der Text für die Bezeichnung eingegeben werden. Das Tool bietet keine Funktion zum vielfachen Einfügen des gleichen Elementes. Überschreitet das Diagramm den zur Verfügung stehenden Zeichenbereich, wird dieser erweitert und eine Scroll-Funktion angeboten. Zur leichteren Verwaltung können Symbole durch Markieren gruppiert werden und somit gemeinsam bearbeitet werden.

Das Tool verfügt weiters über eine Undo-Funktion, mit der die letzten Schritte schnell rückgängig gemacht werden. Im Zeichenbereich ist es möglich Symbole und Elemente zu kopieren und an anderer Stelle wieder einzufügen. Das Löschen der Elemente kann einfach mit der Entf-Taste werden.

1	2	3	4	5	6	7	8	9	10	11	12	13	14
+	+	-	+	+	+	+	+	-	+	+	+	+	+

g) Performance

Der Programmstart von Magic Draw erfolgt trotz Anlegens eines neuen Projektes in einer akzeptablen Zeit. Das Tool zeichnet sich bei der Arbeit, mit einer Hardware welche die Mindestanforderungen erfüllt, durch eine rasche Verarbeitung der verschiedenen Operationen aus. Die Suchezeiten stellen keinen Verzögerungsfaktor für eine zügige Arbeit dar. Auch die Codegenerierung erfolgt in annehmbarer Zeit.

1	2	3	4
+	o	+	+

h) Adaptivität

Die Ansicht des Tools kann entsprechend der Erfahrungen des Benutzers eingerichtet werden. Es ist möglich auf eine Expertenansicht zu wechseln in der alle möglichen Eigenschaften angezeigt werden.

Weiters kann der Benutzer die Angezeigten Symbolleisten definieren, ihre Position definieren und sie bearbeiten. Der Quellcode des Programms wird nicht mitgeliefert.

1	2	3	4	5	6
+	+	+	o	-	-

5.6 ObjectIF

5.6.1. Methodenbasierte Kriterien

a) Modellierung der Objektstruktur – Objektstatik

ObjectIF speichert seine Projekte in einem lokalen Repository. Neue Objekte und Informationen werden in diesem Projektverzeichnis gespeichert. Beim Anlegen eines neuen Projektes wird auf Namensgleichheit geprüft und gegebenenfalls eine Fehlermeldung ausgegeben. Beim Anlegen von neuen Klassen erfolgt auch eine Überprüfung auf Namensgleichheit. Ebenfalls ist es nicht möglich Elemente von Klassen (Attribute, Methoden) gleich zu benennen. Für Attribute und Methoden kann der Datentyp festgelegt werden. Zusätzlich kann für Methoden auch ein Rückgabewert oder ein Wertebereich angegeben werden. Es ist jedoch nicht möglich Vor- bzw. Nachbedingungen zu definieren.

1	2	3	4	5
+	+	+	+	+

b) Modellierung der Systemstruktur – Systemstatik

ObjectIF erlaubt die Darstellung von rekursiven Beziehungen. Einmal erfasste Assoziationen lassen sich im Nachhinein ändern. Diese Änderungen können über die Eigenschaften der jeweiligen Assoziation vorgenommen werden. Klassen können im Systembaum gegliedert bzw. kategorisiert werden. Die Object Constraint Language (OCL) wird von diesem Tool nicht unterstützt.

1	2	3	4
+	+	o	-

c) Dynamisches Objektverhalten – Objektdynamik

In ObjectIF können Zustandsdiagramme und Sequenzdiagramme im Systembaum erstellt werden. Dabei Werden alle Diagrammtypen auf derselben Ebene angelegt. Eine direkte Zuordnung zu einem Klassendiagramm ist somit nicht möglich. Eine Zuordnung kann bestenfalls über die Diagrammbeschreibung erfolgen. Eine Generalisierung kann unabhängig um Objektmodell dargestellt werden.

1	2	3	4	5	6	7
o	o	+	+	-	+	-

d) Zusammenhang zwischen den Modellen

In ObjectIF kann ein Zustandsdiagramm nicht umkehrbar eindeutig mit einem Objekt bzw. einer Klasse verbunden werden. Im Systembaum wird ein Zustandsdiagramm auf gleicher Ebene wie ein Klassendiagramm angezeigt. Unter einer Klasse werden nur Attribute oder Methoden der jeweiligen Klasse angezeigt.

1	2	3	4
-	-	-	-

e) Abstraktionsebenen / Sichten

Attribute und Methoden können bei Klassendiagrammen ein- bzw. ausgeblendet werden. Die Darstellung von Attributen und Methoden auf unterschiedlichem Detaillierungsniveau wird nicht unterstützt. Auch kann keine Einschränkung der Anzeige in Bezug auf die Sichtbarkeit des Objektes gemacht werden. Der Benutzer kann den Detaillierungsgrad jeder Klasse selbst bestimmen

1	2	3	4
+	-	-	o

f) Vorgangsreihenfolge bei der Entwicklung

Das Tool zwingt den Benutzer bei der Erstellung von Klassen und Informationen sich an eine bestimmte Reihenfolge zu halten. Nach dem Anlegen eines Projektes ist es möglich die verschiedenen Diagramme anzulegen. In den Diagrammen müssen zuerst die Objekte angelegt werden, bevor man die entsprechenden Assoziationen erzeugt. Elemente können auch direkt im Systembaum angelegt werden ohne Verwendung im Diagrammeditor. Dieser Editor wird durch Doppelklick auf das jeweilige Diagramm im Systembaum gestartet. Elemente und Objekte die im Diagrammeditor erzeugt werden, werden direkt in dem Systembaum aufgenommen.

1	2	3
+	+	+

g) Unterstützung bei der Methodenanwendung

Für jedes Dokument ist die Erstellung einer Dokumentation im Wordformat möglich. In diesem Dokument gelangt man durch Doppelklick wieder zurück zum jeweiligen Objekt in ObjectIF. Die Notationen orientieren sich an der UML sind jedoch nicht immer 1:1 konform. Es gibt keine Information in Rahmen einer Konsistenz- bzw. Integritätsprüfung.

1	2	3
+	o	-

5.6.2. Funktionale Kriterien

a) Datenhaltung

Das Speichern der Modellierungsinformationen erfolgt im Systembaum. In Form eines Baummenüs kann auf die Daten zugegriffen werden. Wird ein Symbol im Zeichenbereich gelöscht, bleibt es im Systembaum weiterhin vorhanden. Daten können auch direkt im Systembaum manipuliert werden. Diese Änderungen werden im Zeichenbereich übernommen.

Das Tool verfügt über eine Import-/Exportfunktion. Projekte werden gebündelt in einem toolabhängigen Format „.UML“ abgelegt und verwaltet.

1	2	3	4	5
+	+	+	-	+

b) Browsing-Eigenschaften

In ObjectIF kann man mit Rechtsklick im Systembaum nach Diagrammen und Objekten suchen. Eine Suche nach Ähnlichkeiten oder Anfangsbuchstaben wird dabei nicht unterstützt. Die Suche kann mit Filtern bzw. nach bestimmten Kriterien erfolgen, die durch Anhacken im Suchmenü gesetzt werden. Die Ergebnisse einer Suche werden durch Hyperlinks repräsentiert, durch deren Doppelklick man direkt zum entsprechenden Objekt gelangt.

Diagramme und Objekte werden in ObjekIF im Systembaum angezeigt. Änderungen die im Systembaum vorgenommen werden, werden automatisch in den entsprechenden Diagrammen übernommen. Die Darstellung im Systembaum ist jedoch gewöhnungsbedürftig und wird durch die Tatsache erschwert, dass alle Diagramme auf derselben Ebene angezeigt werden.

1	2	3	4	5	6	7	8	9	10	11
+	-	-	+	+	+	+	o	o	o	+

c) Navigation im Systemmodell

Die Navigation in ObjektIF ist durch das Vorhandensein mehrerer Fenster gewöhnungsbedürftig. Erschwert wird die Navigation durch die Tatsache, dass der Systembaum und der Modellierungseditor nicht in einem Fenster angezeigt werden können. Weiters ist es nicht möglich von einem Zustandsdiagramm auf ein Klassendiagramm zu schließen. Attribute und Methoden der Klassen sind zugleich Hot-Links mit denen man die Eigenschaften eines Objektes aufrufen kann.

1	2	3
o	-	+

d) Konsistenzprüfung / Integrität

Mit ObjectIF sind Assoziationen ins Leere nicht möglich. Weiters ist es nicht möglich, rekursive Vererbungen zu erstellen. In einem Paket kann man nicht mehrere Klassen mit demselben Namen anlegen. Auch in einer Klasse werden die Bezeichnungen der einzelnen Elemente auf Namensgleichheit überprüft.

Das System prüft nicht auf Ähnlichkeiten zwischen Bezeichnungen. Eine Konsistenzprüfung kann automatisch durchgeführt werden. Nach dem Löschen eines Elementes in der Zeichenebene bleibt das Element weiter in Systembaum verfügbar. Mit dem Model-Reviewer kann ein Modell auf verschiedene Kriterien untersucht werden.

1	2	3	4	5	6
+	+	+	+	+	+

e) Report-Ausgabe

ObjectIF unterstützt keine automatisierte Generierung von Reports.

1	2	3	4	5	6	7	8
-	-	-	-	-	-	-	-

f) Datenbankschema-Ausgabe

Mit ObjectIF ist es nicht möglich UML Diagramme in ein Datenbankschema zu transformieren. Diese Funktionalität wird von dem Tool nicht angeboten.

1	2	3	4	5
-	-	-	-	-

g) Diagramm-Ausgabe

Die Diagrammausgabe von ObjectIF ist eher rudimentär gehalten, beinhaltet jedoch alle wesentlichen Informationen. Der Ausdruck entspricht der Darstellung im Modellierungseditor und kann auf mehreren Seiten erfolgen. Das Abstraktionsniveau für den Ausdruck kann bei den Druckeinstellungen festgelegt werden. Hierbei kann die Anzeige von Attributen oder Methoden ausgeblendet werden.

1	2	3	4	5
+	+	+	+	+

h) Offenheit – Import / Export

Das Tool verfügt über die Möglichkeit zusätzliche Add-Ins vom Hersteller zu installieren. Hierfür gibt es einen eigenen Add-In Manager. Im Systembaum und im Modellierungseditor können Texte mittels „Copy and Paste“ kopiert und eingefügt werden.

Alle Projekte werden lokal in Projektordnern gespeichert. Das Speicherformat der einzelnen Dateien ist „.UML“.

1	2	3	4	5
+	o	-	-	-

i) Codegenerierung

Die automatische Codegenerierung wird nur für Java, C#, C++, BPEL, XSD und WSDL unterstützt. Für jede im Modell vorkommende Klasse wird eine entsprechende Codierung vorgenommen. Attribute und Typen werden übernommen. Methodengerüste und eventuelle Zugriffsbeschränkungen werden automatisch erstellt. Generierter Code kann im Nachhinein ergänzt und erweitert werden.

1	2	3	4	5	6	7	8	9	10	11
+	+	+	+	+	+	+	o	-	-	-

j) Reverse-Engineering

Reverse Engineering ist für Java, C#, C++, .Net Projektmappen, BPEL, XSD und WSDL möglich. Es kann auch Code der nicht mit ObjektIF erstellt wurde eingelesen werden. Weiters kann der Code jederzeit mit einem Editor erweitert werden. Die Anpassung der Diagramme erfolgt automatisch.

1	2	3
+	o	+

k) Test / Verifikation

Das Tool ermöglicht das Einbinden von JUnit- und JUnit-Framework. Mit diesem Framework ist es möglich Testfälle zu definieren und zu verifizieren.

1	2
+	-

l) Maßzahlen und Statistiken

ObjectIF bietet mit dem Modell-Reviewer die Möglichkeit Statistiken über die Anzahl verschiedenster Komponenten zu erstellen. Die Berechnung von Maßzahlen ist kein Teil dieses Features.

1	2	3
-	+	-

m) Multi-User – Eigenschaften

ObjectIF unterstützt einen Mehrbenutzerbetrieb. In der Standalone Version ist kein Berechtigungsmodell für Benutzer implementiert. Beim Öffnen eines Projektes kann ein Benutzer festlegen, ob das entsprechende Projekt für andere Benutzer gesperrt ist, oder ob mehrere Benutzer gleichzeitig auf das Projekt zugreifen können.

1	2	3	3
+	-	+	o

n) Security

Das Tool als solches unterstützt keine Benutzerverwaltung. Der Benutzer braucht sich nach dem Programmstart nicht anzumelden. Eine solche Sicherheitsfunktion ist nicht implementiert. Für jedes Projekt oder Diagramm kann der letzte Benutzer ausgelesen werden. Das Tool verfügt weiters über eine automatische Sicherungsfunktion. Sicherungseinstellungen können in den entsprechenden Einstellungen vorgenommen werden. Bei Falscheingaben wird die Systemoberfläche eingefroren und der Benutzer erhält eine Fehlermeldung.

Das Löschen von Modellierungsinformationen muss vom Benutzer ein zweites Mal bestätigt werden. ObjectIF verfügt über keine Undo-Funktion. Somit können einmal gelöschte Objekte nicht wieder hergestellt werden sofern die Änderungen gespeichert wurden.

1	2	3	4	5	6	7	8	9
-	-	o	+	+	o	+	+	-

o) Multi-Projekt – Eigenschaften / Versionsmanagement

Mit dem Tool können gleichzeitig verschiedene Projekte geöffnet bzw. bearbeitet werden. Dafür muss das Tool jedoch mehrfach gestartet werden. Der Austausch von Modellierungsinformationen zwischen verschiedenen Projekten kann über eine Erstellung aus einem bestehenden Projekt erfolgen. Eine Versionierung wird bei der Standalone Version des Tools nicht unterstützt.

1	2	3	4	5
o	+	+	-	-

5.6.3. Ergonomie

a) Benutzeroberfläche

Das Erscheinungsbild von ObjectIF ist vor allem zu Beginn sehr gewöhnungsbedürftig. Anders als bei anderen Tools, muss für ObjectIF für jede Grundfunktionalität ein eigenständiges Fenster geöffnet werden. Das Aussehen von ObjectIF besitzt keine Ähnlichkeit mit bekannten Windows-Programmen. Die GUI ist nicht für eine intuitive Verwendung ausgelegt. Auch die Namensgebung im Hauptfenster lässt nicht intuitiv auf mögliche Funktionalitäten schließen. Die Werkzeuge im

Modelleditor sind übersichtlich angeordnet. Die Dialoge und Eingabemasken sind einfach zu bedienen. Momentan nicht zur Verfügung stehende Menüpunkte werden ausgegraut. Bei längeren Ladezeiten für Aktionen gibt es keinen Fortschrittsbalken. Die Anzeige der verschiedenen Listen ist einheitlich gehalten.

1	2	3	4	5	6	7
o	o	o	o	+	-	+

b) Tool-Dokumentation

Mit der Installation von ObjectIF wird auch ein Benutzerhandbuch auf dem Rechner installiert, das direkt im Programm aufgerufen werden kann. Darüber hinaus verfügt das Programm über eine Kurzfassung der Grundlagen für einen schnellen Einstieg. Die einzelnen Beschreibungen werden durch Screenshots ergänzt, um ein besseres und leichteres Verständnis zu schaffen. Weiters bietet das Tool Tutorials für verschiedene Anwendungsfälle. Zusätzlich wird noch eine Online Hilfe angeboten, welche nach Inhalten gegliedert ist. Links zur Herstellerhomepage und anderen Informationsquellen können ebenfalls direkt im Tool aufgerufen werden.

Die Dokumentationen sind alle gut gegliedert und strukturiert, verfügen über ein Inhaltsverzeichnis und in sind in der deutschen Sprache verfügbar. Weitere Hilfestellungen findet der Benutzer auf der Herstellerhomepage in Form von PDF-Dokumenten, die man auf seinen Rechner herunterladen kann.

1	2	3	4	5	6	7	8	9	10	11	12
+	+	+	+	+	+	-	+	+	+	+	-

c) Usability

Leider ermöglicht das Tool kein rasches und effizientes Arbeiten. Im Fehlerfall werden einfache Meldungen ausgeben, ohne eine detaillierte Erklärung. Das Tool unterstützt nicht die Verwendung von Tastatur-Shortcuts. Bei der Eingabe von ungültigen Werten wird nicht immer eine Fehlermeldung ausgegeben. In gewissen Fällen kommt es vor, dass bei einer Falscheingabe einfach der Bestätigungsbutton ausgegraut wird. Eine Symbolleiste wird nur im Modellierungseditor in Form einer Werkzeugliste angezeigt. Diese Werkzeugliste kann nicht angepasst werden. Das Tool verfügt weiters über keine Statusleiste für die Anzeige des Verarbeitungsstatus. Eine Mehrfenstertechnik für ein Projekt ist nicht implementiert. In den angezeigten Listen ist auch keine Mehrfachauswahl möglich. ObjectIF ist in englischer und deutscher Sprache verfügbar.

1	2	3	4	5	6	7	8	9	10	11
o	o	-	-	-	o	o	-	-	+	+

d) Hilfefunktionen

Die Hilfefunktion ist nur über das Menü erreichbar und kann nicht über die Funktionstaste F1 aufgerufen werden. Zusätzlich zum Handbuch im PDF-Format verfügt das Tool über eine Online Hilfe, die das Suchen nach bestimmten Themen und Inhalten ermöglicht. Eine Hot-Help Funktion für die verschiedenen Symbole ist nicht implementiert.

1	2	3	4
-	+	+	-

e) Graphikeditor / Darstellung

Der Modellierungseditor von ObjectIF ist sehr einfach und rudimentär gehalten. Der Benutzer kann so gut wie keine individuellen Anpassungen vornehmen. Zusätzliche Objekte und Modellelemente, die nicht von dem jeweiligen Diagrammtyp benötigt werden, stehen nicht zur Verfügung. Für Klassen besteht die Möglichkeit die Füllfarbe zu ändern. Für Assoziationen können z.B. keine Anpassungen vorgenommen werden.

Die am Bildschirm dargestellten Diagramme entsprechen dem späteren Ausdruck. ObjectIF besitzt eine Zoomfunktion, die über die Werkzeugliste aufgerufen wird. Symbole und Texte können nicht rotiert werden.

Assoziationen können durch nachträglich festgelegte Eckpunkte verändert werden. Die Strichstärke und Linienart kann man nicht verändern. Ein Refresh ist nicht vorhanden.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
o	-	o	o	-	+	-	-	+	-	+	-	+	-	-

f) Graphikeditor / Bedienung

Im Modellierungseditor können Notationselemente einfach durch Ziehen von der Werkzeugliste in den Zeichenbereich erstellt werden. Sofort nach dem Anlegen kann die entsprechende Bezeichnung in Textform eingeben werden. Weiters verfügt das Tool über eine Haltefunktion. Wählt man in der Werkzeugliste z.B. eine Klasse aus, kann man im Zeichenbereich so viele Klassen erstellen wie man benötigt, ohne nochmals das entsprechende Element auswählen zu müssen.

Der Zeichenbereich für Diagramme ist unendlich groß. Im Zeichenbereich können mehrere Symbole gleichzeitig markiert und verschoben werden. Das Tool bietet keine Undo- und Redo-Funktion. Ebenso ist es im Modellierungseditor nicht möglich Objekte zu kopieren und wieder einzufügen. Um Symbole zu löschen muss man einen Rechtsklick auf das entsprechende Symbol machen und Löschen auswählen. Symbole können nicht durch Drücken der Entfernen-Taste gelöscht werden.

1	2	3	4	5	6	7	8	9	10	11	12	13	14
+	+	+	+	+	+	+	+	-	-	-	+	-	-

g) Performance

ObjectIF ist in Bezug auf Arbeitsgeschwindigkeit ein durchaus akzeptables Tool. Beim Zoomen und Scrollen kommt es zu keinen wesentlichen Zeitverzögerungen. Auch die Suche, welche zwar abhängig vom Suchspektrum ist, erfolgt ohne größere Verzögerungen. Einzig das Laden neuer Projekt nimmt etwas Zeit in Anspruch. Auch die Zeit bei der Codegenerierung weicht nicht wesentlich von anderen Tools wie z.B. Magic Draw ab.

1	2	3	4
+	o	+	+

h) Adaptivität

Bezüglich der Adaptivität schneidet ObjectIF ähnlich wie Poseidon for UML nicht gut ab. Dieses Produkt stellt keine Funktionen zur Verfügung, entsprechend des Lernfortschritts des Anwenders die Menü- und Symbolleisteneinträge zu verändern und in ihrem Umfang anzupassen.

Auch der Quellcode des Programms wird nicht mitgeliefert.

1	2	3	4	5	6
-	-	-	-	-	-

6 Empirische Bewertung der Werkzeug-Bedienung

In diesem Kapitel wird analysiert, wie schnell und effizient ein ausgewähltes Projekt erstellt werden kann. Zur Anwendung kommen die UML-Werkzeuge StarUML, Poseidon for UML, Objecteering, Visual Paradigm, Magic Draw und ObjectIF. Es wird die Modellierung anhand verschiedener UML-Diagrammtypen betrachtet.

Als Basisidee zur empirischen Bewertung der Werkzeug-Bedienung dient das GOMS-Verfahren.

6.1 GOMS

GOMS bezeichnet Methoden zur Aufgabenanalyse in einer Mensch-Computer-Umgebung. GOMS setzt sich aus vier Hauptkomponenten zusammen: Goals, Operators, Methods und Selection Rules. GOMS-Modelle dienen zur Analyse von Aufgaben und zur Vorhersage über die Wege zur Erfüllung und über die Dauer einer Aufgabenbearbeitung.

Ziel von solchen Modellen, die die Modellierung von Nutzerverhalten unterstützten, ist die Kosten-, Zeit- und Aufwandersparnis bei der Entwicklung von Interfaces in der Computer-Mensch-Interaktion.

Bei der Modellierung werden die Benutzeraktionen dargestellt, die zur Erledigung der Aufgaben nötig sind. Daraus können auch Vorhersagen über die Dauer von bestimmten Aktionen abgeleitet werden.

Unter **Goals** versteht man Aufgabenziele und untergeordnete Ziele, die sich aus der Analyse der Aufgabe ergeben. Im klassischen GOMS sind diese hierarchisch angeordnet. Zum besseren Verständnis soll das folgende Beispiel dienen.

Hierbei handelt es sich um ein elektronisches Dokument, das Rechtschreibfehler enthält. Diese sind mit roter Farbe gekennzeichnet. Ziel ist es, die markierten Stellen zu korrigieren.

```
GOAL: EDIT-DOCUMENT
        GOAL: EDIT-UNIT-TASK repeat until no more unit tasks
                GOAL: ACQUIRE-UNIT-TASK
                GOAL: EXECUTE-UNIT-TASK
```

Hierbei ist das übergeordnete Ziel EDIT-DOCUMENT. Dieses Ziel wird in weitere Ziele unterteilt. Ein Unterziel ist EDIT-UNIT-TASK, das Korrigieren einer markierten Stelle. Dies soll so lange wiederholt werden, bis es keine weiteren Verbesserungen mehr gibt. EDIT-UNIT-TASK wird weiter aufgespaltet in ACQUIRE-UNIT-TASK und EXECUTE-UNIT-TASK.
Ein Ziel kann nicht erreicht werden, bis alle Unterziele erfolgreich erreicht wurden. Das Dokument ist also nicht fertig, bis alle markierten Stellen gefunden und korrigiert wurden.

Operatoren sind die Aktionen des Benutzers, um die Zustände einer Aufgabenbearbeitung zu ändern. Operatoren haben eine bestimmte Dauer. Deshalb ist es auch

möglich, die Dauer der Bearbeitung einer Aufgabe vorherzusagen. Die Dauer der Aufgabe ist abhängig von der Anzahl der benötigten Operatoren. Die Dauer von bestimmten Aktionen erfolgt durch Beobachtung und Analyse. Eine Beobachtung kann die Dauer des Wechselns der Hand von der Tastatur zur Maus sein.

```
GOAL: EDIT-DOCUMENT
        GOAL: EDIT-UNIT-TASK repeat until no more unit tasks
                GOAL: ACQUIRE-UNIT-TASK
                        GET-NEXT-PAGE at the end of the page
                        GET-NEXT-TASK
                GOAL: EXECUTE-UNIT-TASK
```

GET-NEXT-PAGE ist die Aktion, wenn der Benutzer am Ende einer Seite angelangt ist. Er wechselt auf die nächste Seite. GET-NEXT-TASK ist die nächste Aktion, die der Benutzer erledigen soll, das Korrigieren der weiteren Fehler.

Methods sind hinterlegte, bzw. gespeicherte und verifizierte Abfolgen von Operatoren und Zielen. Methoden besitzen die Eigenschaften, dass sie sicher zum Erfolg führen und sie stehen zur Laufzeit zur Verfügung.
Die Methoden sind bekannt, ihre Wirkung ist bekannt und sind für die Erreichung der Ziele notwendig.

```
GOAL: ACQUIRE-UNIT-TASK
        GET-NEXT-PAGE ist at the end of the page
        GET-NEXT-TASK
```

Diese Folge ist eine Methode, mit der das Ziel ACQUIRE -UNIT-TASK immer erreicht wird.

Mit **Selection Rules**, zu Deutsch „Auswahlregeln“, werden Entscheidungen getroffen, welche Methoden angewandt werden.

```
GOAL: EXECUTE-UNIT-TASK
        GOAL: LOCATE-LINE
                [select: USE-LF-METHOD
                         USE-QS-METHOD]
```

Um das Ziel LOCATE-LINE zu erfüllen, d.h. die Zeile zu finden, in der die nächste Korrektur gemacht werden soll, gibt es zwei Möglichkeiten. Mit der Methode USE-LF-METHOD wird Zeile für Zeile durchgesucht (LineFeed) bis die entsprechende Stelle gefunden wurde. Die USE-QS-METHOD ist die zweite Möglichkeit, mit der über eine String-Suche die richtige Zeile gefunden werden kann. Abhängig von der Umwelt kann man Rückschlüsse zur Wahl der Methode machen. Ist eine Zeile nur wenige Zeilen von der aktuellen Stelle entfernt, wird der Benutzer wohl die LineFeed-Methode benutzen. Ist die nächste zu korrigierende Stelle weiter entfernt, ist die QuotedString-Methode wahrscheinlicher.[3]

[3] Vgl. http://www.eecs.umich.edu/~kieras/goms.html (Stand: 17.04.2007)

Im nächsten Abschnitt werden die beiden UML-Werkzeuge anhand der Grundkonzepte von GOMS verglichen. Es wurden die benötigten Aktionen für das Erreichen von Zielen, hier das Modellieren von verschiedenen Diagrammen, mitprotokolliert. In der ersten Spalte in der unteren Tabelle sind die Ziele definiert, in der zweiten Spalte die Methoden und in der dritten Spalte ist die Anzahl der Operatoren für die Erfüllung des Zieles angegeben.

6.2 Beispielprojekt: Onlinebuchgeschäft

Zur Bewertung der Bedienung wurde ein Ausschnitt eines Modells eines Onlinegeschäftes erstellt, das im Bereich des Internethandels von Büchern tätig ist:

Benutzer, die über ein Benutzerkonto verfügen, können Einkäufe im Onlinebuchgeschäft tätigen. Bevor ein Kunde mit dem System interagieren kann, muss er sich am System mit seinem Benutzernamen und Passwort anmelden. Vom Benutzer sind der Name, Vorname, sowie die Adresse zu speichern. Ein Benutzer kann auch seine persönlichen Informationen im System ändern. Weiters existieren zwei Arten der Bezahlung: die Bezahlung per Überweisung oder mit Kreditkarte. Neben der Kontonummer und der Bezeichnung des Kontoinhabers werden noch weitere Informationen je nach Art des Zahlungsvorgangs benötigt. Das sind der Bankname und die Bankleitzahl für die Bezahlung per Kontoüberweisung bzw. die Kartennummer, das Gültigkeitsdatum der Kreditkarte und die Bezeichnung des Kreditinstitutes.

Eine Bestellung enthält mindestens ein Buch. Von der Bestellung sind die Bestellnummer, das Bestelldatum, die Bestellsumme und die Anzahl der bestellten Bücher von Interesse. Bücher werden von Verlagen verlegt und können in ein Sortiment eingeordnet werden. Von den Büchern sind der Titel, die ISBN-Nummer, der Autor, der Preis, eine Kurzbeschreibung und der Steuersatz zu speichern.

6.3 Beschreibung der Analyse

Bei der Analyse werden die einzelnen Benutzerinteraktionen mitgezählt. Dabei handelt es sich um Mausklicks, um Tastatureingaben und um das Wechseln von der Tastatur zur Maus bzw. von der Maus zur Tastatur. Zu den Mausklicks zählen der einfache bzw. der Doppelklick, sowie drag-and-drop Aktivitäten.

Die hierbei verwendeten Abkürzungen:

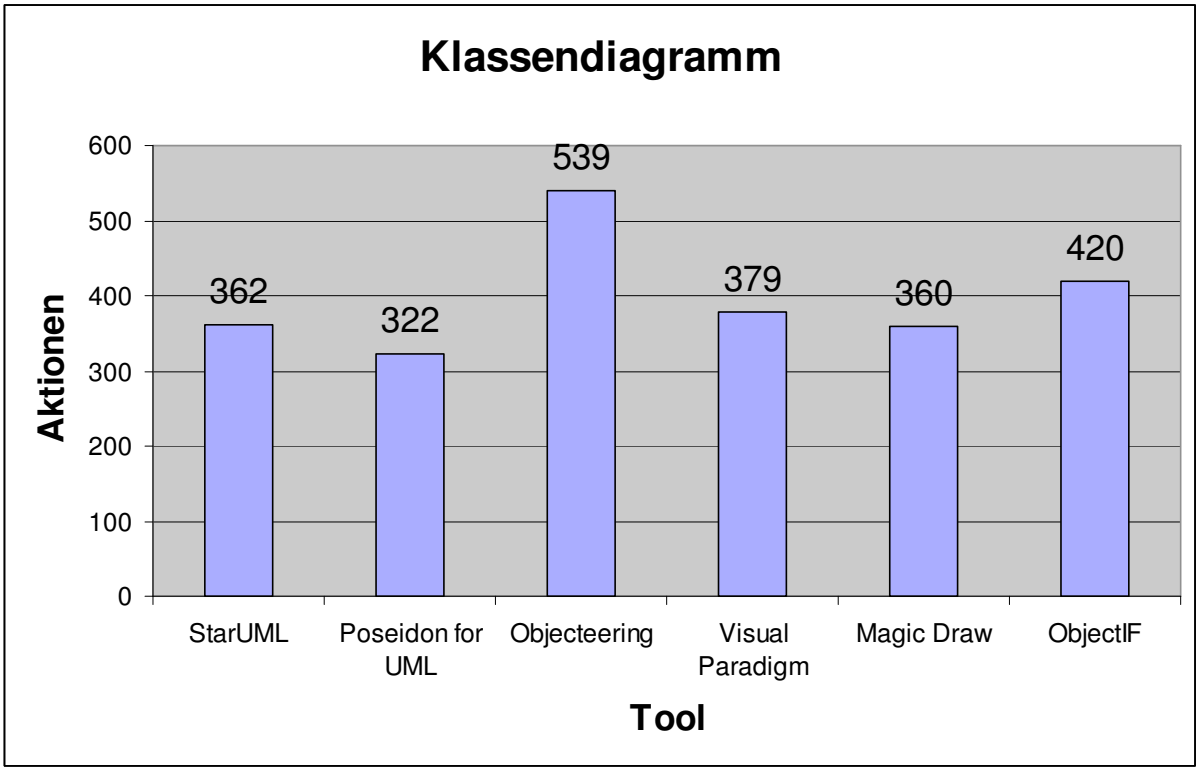
LK	Klick mit der linken Maustaste
RK	Klick mit der rechten Maustaste
DK	Doppelklick mit der linken Maustaste
WE	Wechseln zwischen den Eingabegeräten

Die erste Spalte der Auswertungstabelle beschreibt die einzelnen Modellierungsschritte, die zweite Spalte gibt Auskunft über die Vorgangsweise der Modellierung und in der dritten Spalte werden die jeweils benötigten Aktionen gezählt.

Das zugrunde liegende Modell beschreibt nur einen Ausschnitt des realen Problembereichs, da die entstandenen Diagramme für die Art der Auswertung nicht übermäßig komplex sein müssen. Weiters wurden die einzelnen Aktionen (Mausklicks, Tastatureingaben und das Wechseln zwischen diesen beiden Eingabegeräten) in ihrer Wertigkeit gleichgesetzt. Natürlich spielen bei der Erstellung eines Diagramms auch die Länge der Mauswege und der Aufbau der einzelnen Menüs und Dialogfenster eine Rolle, die bei dieser Messung nicht betrachtet wurden.

6.4 Erstellung des Klassendiagramms

6.4.1. Zusammenfassung der Ergebnisse



6.4.2. Interpretation der Ergebnisse

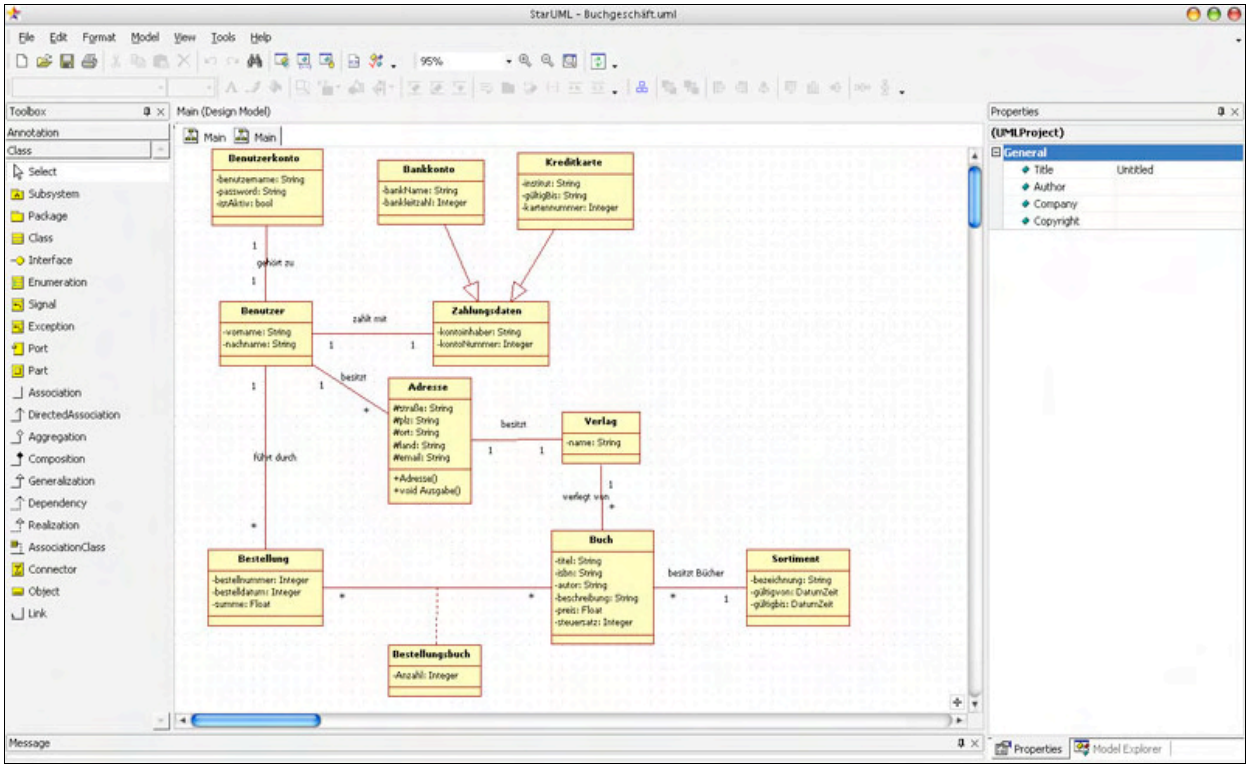
Ein Grund weshalb Poseidon for UML bei der Gestaltung des Klassendiagramms, klar weniger Aktionen benötigt, ist die deutlich bessere Bedienbarkeit der einzelnen Werkzeuge. Will man mit StarUML ein Element in das Diagramm einfügen, muss das für diesen Vorgang benötigte Werkzeug zuerst aus Menüleiste gewählt werden. Erst dann kann der Benutzer die gewünschte Aktion durchführen. Bei Poseiden for UML braucht man z.B. beim Hinzufügen einer Assoziation zwischen zwei Klassen nur eine der beiden Klassen per Maus „berühren“ und schon wird neben dieser Klasse ein Icon eingeblendet, z.B. das Icon für das Einfügen einer Assoziation. Ein Mausklick auf dieses Icon und der Benutzer kann die gewünschte Assoziation durch das Verbinden zweier Klassen einfügen. Dieses Bedienkonzept findet man bei Poseidon for UML in allen verschiedenen Diagrammen wieder.

Das gleiche Bedienkonzept verwendet auch Visual Paradigm. Das beschleunigt den Entwicklungsprozess, denn der Benutzer erspart sich lange Wege mit der Maus und wiederholte Klicks. Bevor man aber diese Vorteile wirklich nutzen kann, benötigt der Anwender eine Eingewöhnungsphase, denn dieses Bedienkonzept unterscheidet sich zu anderen Entwicklerwerkzeugen mit denen man in der Vergangenheit gearbeitet hat.

ObjectIF, aber vor allem Objecteering zeichnen sich leider durch eine sehr hohe Anzahl von Aktionen aus. Diese entstehen vor allem durch viele Mausklicks. So kann man beispielsweise eine Klasse nicht wie in Magic Draw oder Visual Paradigm einfach in den Zeichenbereich ziehen, sondern man muss 2 Mal klicken (ein Mal auf das Symbol, einmal in den Zeichenbereich). Auch das Anlegen von Assoziationen ist aufwändiger.

Ein weiterer Nachteil von ObjectIF ist, dass einige wichtige Notationselemente fehlen. Einige einfache Notationselemente wie Aggregation und Generalisierung sind nicht UML-konform. ObjectIF ist nicht so intuitiv zu bedienen wie andere Tools und verlangt nach einer längeren Eingewöhnungsphase. Positiv an ObjectIF ist, dass man zum Beispiel beim gleichzeitigen Anlegen mehrerer Klassen nicht immer wieder auf das Klassensymbol klicken muss. Es genügt, ein Mal auf das Symbol zu klicken. Anschließend muss man für jede Klasse nur mehr ein Mal in den Zeichenbereich klicken. Objecteering verlangt zwar nach mehreren Aktionen, ist dafür aber kostenlos erhältlich. Wenn man nicht unbedingt ein Freeware-Produkt sucht, sind für das Design eines Klassendiagramms StarUML, Poseidon for UML, Visual Paradigm und Magic Draw am Besten geeignet.

6.4.3. Klassendiagramm – StarUML

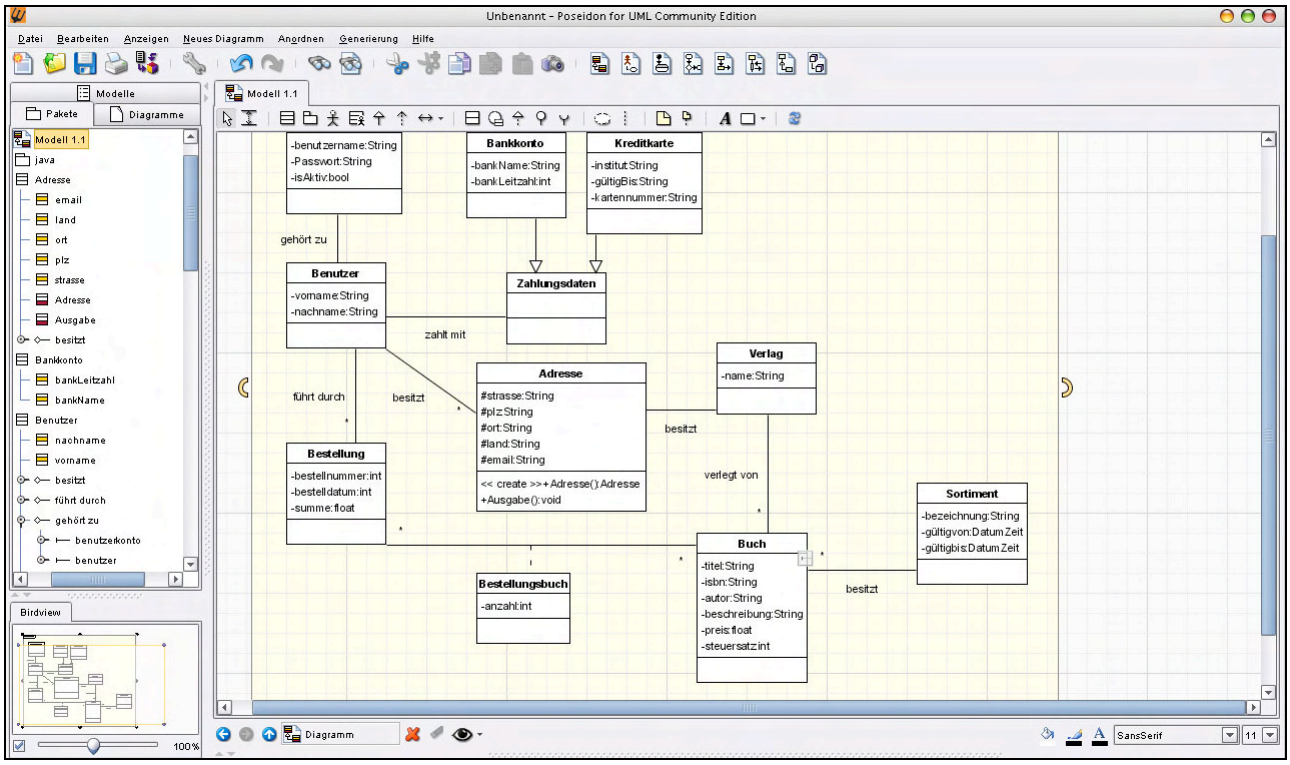


Schritt	Vorgangsweise	Benötigte Aktionen
01 - Neues Projekt anlegen.	Über das Menü *File – New Projekt* und danach Eingabe der Projekteigenschaften unter *Properties*.	Maus: 4 Tastatur: 6 Wechsel: 2 12
02 - Anlegen eines neuen Klassendiagramms im Projekt.	Wechseln in den Modell-Explorer und über die Baumstruktur zum gewünschten Modell.	Maus: 2 Tastatur: 0 Wechsel: 0 2
03 - Anlegen der Klassen: Benutzerkonto, Benutzer, Bestellung, Adresse, Zahlungsdaten, Bankkonto, Kreditkarte, Buch, Verlag, Sortiment und Bestellungsbuch.	Links im Wergzeugsfenster per LK auf das Klassensymbol und danach LK auf die Arbeitsfläche. Tastatureingabe der Klassenbezeichnungen und Bestätigung durch Enter. Diese Aktivitäten werden für alle neun Klassen wiederholt.	Maus: 22 Tastatur: 22 Wechsel: 44 88
04 - Hinzufügen der Attribute der Klasse Benutzerkonto.	DK auf gewünschte Klasse, Klick auf Icon *Add Attribut,* Eingabe der Attributbezeichnung sowie des Typs. Setzen der Sichtbarkeit	Maus: 9 Tastatur: 3 Wechsel: 6 18

Schritt	Vorgangsweise	Benötigte Aktionen
05 - Hinzufügen der Attribute der Klasse Adresse.	Analog zu Schritt 04.	Maus: 15 Tastatur: 5 Wechsel: 10 30
06 - Hinzufügen der Attribute der Klasse Benutzer.	Analog zu Schritt 04.	Maus: 6 Tastatur: 2 Wechsel: 4 12
07 - Hinzufügen der Attribute der Klasse Zahlungsdaten.	Analog zu Schritt 04.	Maus: 6 Tastatur: 2 Wechsel: 4 12
08 - Hinzufügen der Attribute der Klasse Bankkonto.	Analog zu Schritt 04.	Maus: 6 Tastatur: 2 Wechsel: 4 12
09 - Hinzufügen der Attribute der Klasse Kreditkarte.	Analog zu Schritt 04.	Maus: 9 Tastatur: 3 Wechsel: 6 18
10 - Hinzufügen der Attribute der Klasse Buch.	Analog zu Schritt 04.	Maus: 18 Tastatur: 6 Wechsel: 12 36
11 - Hinzufügen der Attribute der Klasse Verlag.	Analog zu Schritt 04.	Maus: 3 Tastatur: 1 Wechsel: 2 6
12 - Hinzufügen der Attribute der Klasse Sortiment.	Analog zu Schritt 04.	Maus: 9 Tastatur: 3 Wechsel: 6 18
13 - Hinzufügen der Attribute der Klasse Bestellungsbuch.	Analog zu Schritt 04.	Maus: 3 Tastatur: 1 Wechsel: 2 6
14 - Anlegen einer Assoziation zwischen den Klassen Benutzerkonto und Benutzer.	LK auf das Symbol *Assoziation* und Verbinden der beiden Klassen durch Benutzung der Maus. Angabe der Muliplizitäten durch Listenauswahl und Benennung der Assoziation.	Maus: 9 Tastatur: 1 Wechsel: 2 12
15 - Anlegen einer Assoziation zwischen den Klassen Benutzer und Bestellung.	Analog zu Schritt 14.	Maus: 9 Tastatur: 1 Wechsel: 2 12

Schritt	Vorgangsweise	Benötigte Aktionen
16 - Anlegen einer Assoziation zwischen den Klassen Benutzer und Adresse.	Analog zu Schritt 14.	Maus: 9 Tastatur: 1 Wechsel: 2 12
17 - Anlegen einer Assoziation zwischen den Klassen Benutzer und Zahlungsdaten.	Analog zu Schritt 14.	Maus: 9 Tastatur: 1 Wechsel: 2 12
18 - Anlegen einer Assoziation zwischen den Klassen Adresse und Verlag.	Analog zu Schritt 14.	Maus: 9 Tastatur: 1 Wechsel: 2 12
19 – Anlegen einer Assoziation zwischen den beiden Klassen Bestellung und Buch.	Analog zu Schritt 14.	Maus: 9 Tastatur: 1 Wechsel: 2 12
20 - Anlegen einer Assoziation zwischen den Klassen Verlag und Buch.	Analog zu Schritt 14.	Maus: 9 Tastatur: 1 Wechsel: 2 12
21 - Anlegen einer Assoziation zwischen den Klassen Buch und Sortiment.	Analog zu Schritt 14.	Maus: 9 Tastatur: 1 Wechsel: 2 12
22 - Verbinden der Assoziationsklasse Bestellungsbuch und der Assoziation.	LK auf das Symbol *AssociationClass* und verbinden der Klasse mit der Assoziation.	Maus: 2 Tastatur: 0 Wechsel: 0 2
23 - Einfügen der beiden Generalisierungen zwischen den Klassen Zahlungsdaten und Bankkonto bzw. Zahlungsdaten und Kreditkarte.	LK auf Generalisierung und verbinden der jeweiligen Klassen.	Maus: 4 Tastatur: 0 Wechsel: 0 4
GESAMT		**362**

6.4.4. Klassendiagramm - Poseidon for UML



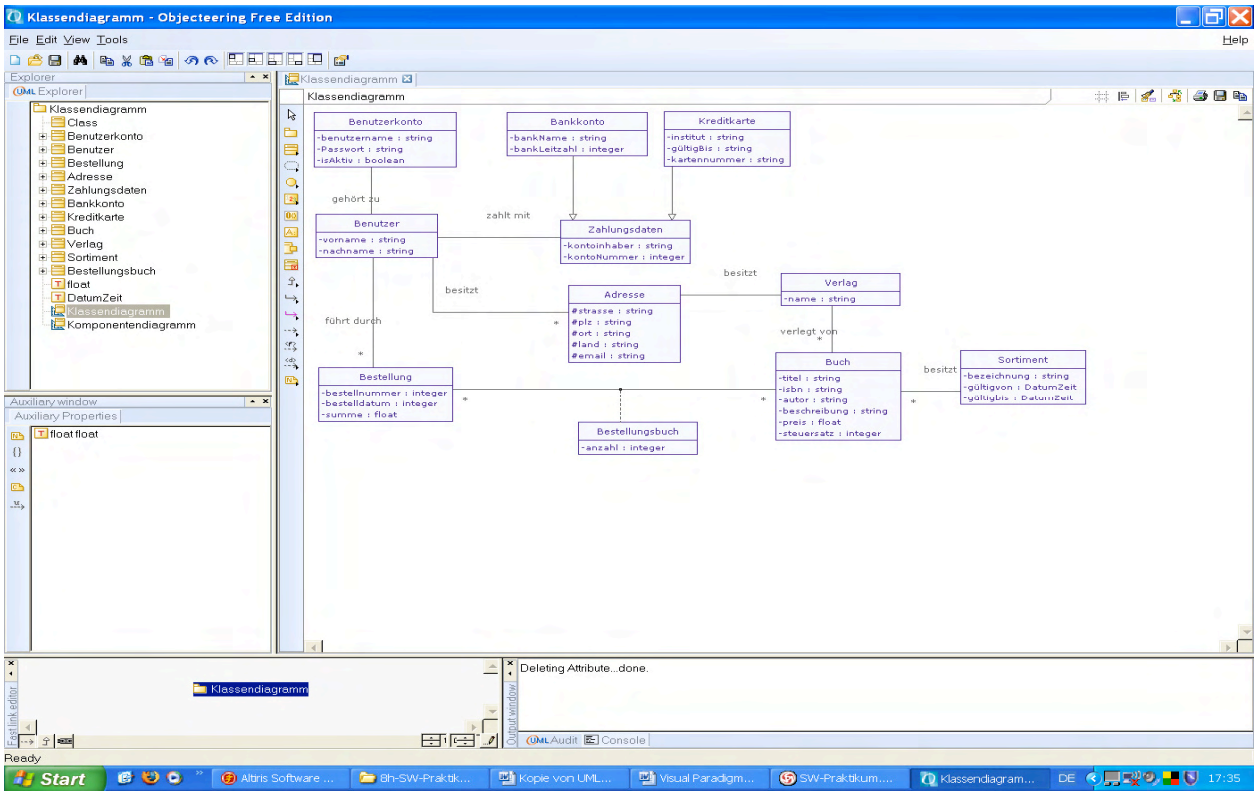
Schritt	Vorgangsweise	Benötigte Aktionen
01 - Neues Projekt anlegen	Nach dem Start des Programms wird automatisch ein neues Projekt angelegt. Name des Projektes kann unter den Projekteigenschaften geändert werden.	Maus: 1 Tastatur: 1 Wechsel: 2
		4
02 - Anlegen eines neuen Klassendiagramms.	Ein Klassendiagramm wird nach dem Programmstart automatisch angelegt. Ändern der Diagrammeigenschaften über das Karteikartensystem.	Maus: 1 Tastatur: 1 Wechsel: 2
		4
03 - Anlegen der Klassen: Benutzerkonto, Benutzer, Bestellung, Adresse, Zahlungsdaten, Bankkonto, Kreditkarte, Buch, Verlag und Sortiment.	In der Wergzeugleiste, per LK auf das Klassensymbol und danach LK auf die Arbeitsfläche. Eingabe der Klassenbezeichnungen und Bestätigung durch Enter. Diese Aktivitäten werden für alle neun Klassen wiederholt.	Maus: 20 Tastatur: 20 Wechsel: 40
		80

Schritt	Vorgangsweise	Benötigte Aktionen
04 - Hinzufügen der Attribute der Klasse Benutzerkonto.	Mausnavigation über gewünschte Klasse, Klick auf Icon *Attribut hinzufügen,* Eingabe der Attributbezeichnung, des Typs und setzten durch LK der Sichtbarkeit unter Eigenschaften.	Maus: 6 Tastatur: 3 Wechsel: 6 15
05 - Hinzufügen der Attribute der Klasse Benutzer.	Analog zu Schritt 04.	Maus: 4 Tastatur: 2 Wechsel: 4 10
06 - Hinzufügen der Attribute der Klasse Bestellung.	Analog zu Schritt 04.	Maus: 6 Tastatur: 3 Wechsel: 6 15
07 - Hinzufügen der Attribute der Klasse Adresse.	Analog zu Schritt 04.	Maus: 6 Tastatur: 3 Wechsel: 6 15
08 - Hinzufügen der Attribute der Klasse Zahlungsdaten.	Analog zu Schritt 04.	Maus: 4 Tastatur: 2 Wechsel: 4 10
09 - Hinzufügen der Attribute der Klasse Bankkonto.	Analog zu Schritt 04.	Maus: 4 Tastatur: 2 Wechsel: 4 10
10 - Hinzufügen der Attribute der Klasse Kreditkarte.	Analog zu Schritt 04.	Maus: 6 Tastatur: 3 Wechsel: 6 15
11 - Hinzufügen der Attribute der Klasse Buch.	Analog zu Schritt 04.	Maus: 12 Tastatur: 6 Wechsel: 12 30
12 - Hinzufügen der Attribute der Klasse Verlag.	Analog zu Schritt 04.	Maus: 2 Tastatur: 1 Wechsel: 2 5
13 - Hinzufügen der Attribute der Klasse Sortiment.	Analog zu Schritt 04.	Maus: 6 Tastatur: 3 Wechsel: 6 15
14 - Anlegen der Methoden der Klasse Adresse.	DK auf die gewünschte Klasse, Hinzufügen von Methoden über Icon *Methode hinzufügen* und Eingabe der Bezeichnung per Tastatur, Setzen der Sichtbarkeit unter durch LK unter Eigenschaften.	Maus: 4 Tastatur: 2 Wechsel: 4 10

Schritt	Vorgangsweise	Benötigte Aktionen
15 - Anlegen einer Assoziation zwischen den Klassen Benutzerkonto und Benutzer.	LK auf das Icon *Assoziation hinzufügen* und Verbinden der beiden Klassen durch Benützung der Maus. Benennung der Assoziation durch DK und Tastatureingabe. Ändern der Multiplizitäten durch DK und Eingabe über Listenfeld.	Maus: 6 Tastatur: 1 Wechsel: 2 8
16 - Anlegen einer Assoziation zwischen den Klassen Benutzer und Zahlungsdaten.	Analog zu Schritt 15.	Maus: 6 Tastatur: 1 Wechsel: 7 14
17 - Anlegen einer Assoziation zwischen den Klassen Benutzer und Adresse.	Analog zu Schritt 15.	Maus: 6 Tastatur: 1 Wechsel: 2 9
18 - Anlegen einer Assoziation zwischen den Klassen Benutzer und Bestellung.	Analog zu Schritt 15.	Maus: 6 Tastatur: 1 Wechsel: 2 9
19 - Anlegen einer Assoziation zwischen den Klassen Verlag und Buch.	Analog zu Schritt 15.	Maus: 6 Tastatur: 1 Wechsel: 2 9
20 - Anlegen einer Assoziation zwischen den Klassen Buch und Sortiment.	Analog zu Schritt 15.	Maus: 6 Tastatur: 1 Wechsel: 2 9
21 - Anlegen einer Assoziation zwischen den Klassen Bestellung und Buch.	Analog zu Schritt 15.	Maus: 6 Tastatur: 1 Wechsel: 2 9
22 - Anlegen einer Assoziation zwischen den Klassen Adresse und Verlag.	Analog zu Schritt 15.	Maus: 6 Tastatur: 1 Wechsel: 2 9
23 - Anlegen der Assoziationsklasse Bestellungsbuch mit deren Attribut.	LK auf das Assoziationen Drop-Down Menü und LK auf *Assoziationsklasse einfügen*. Ziehen der Assoziationsklasse im Klassendiagramm und Benennung der Klasse. Danach Hinzufügen des Attributs per LK.	Maus: 7 Tastatur: 2 Wechsel: 4 13

Schritt	Vorgangsweise	Benötigte Aktionen
24 - Hinzufügen der Generalisierung für die Klassen Bankkonto und Kreditkarte mit der Klasse Zahlungsdaten.	LK auf das Symbol *Generalisierung* und Verbinden der beiden Klassen durch Benützung der Maus.	Maus: 4 Tastatur: 0 Wechsel: 0
		4
GESAMT		**322**

6.4.5. Klassendiagramm – Objecteering



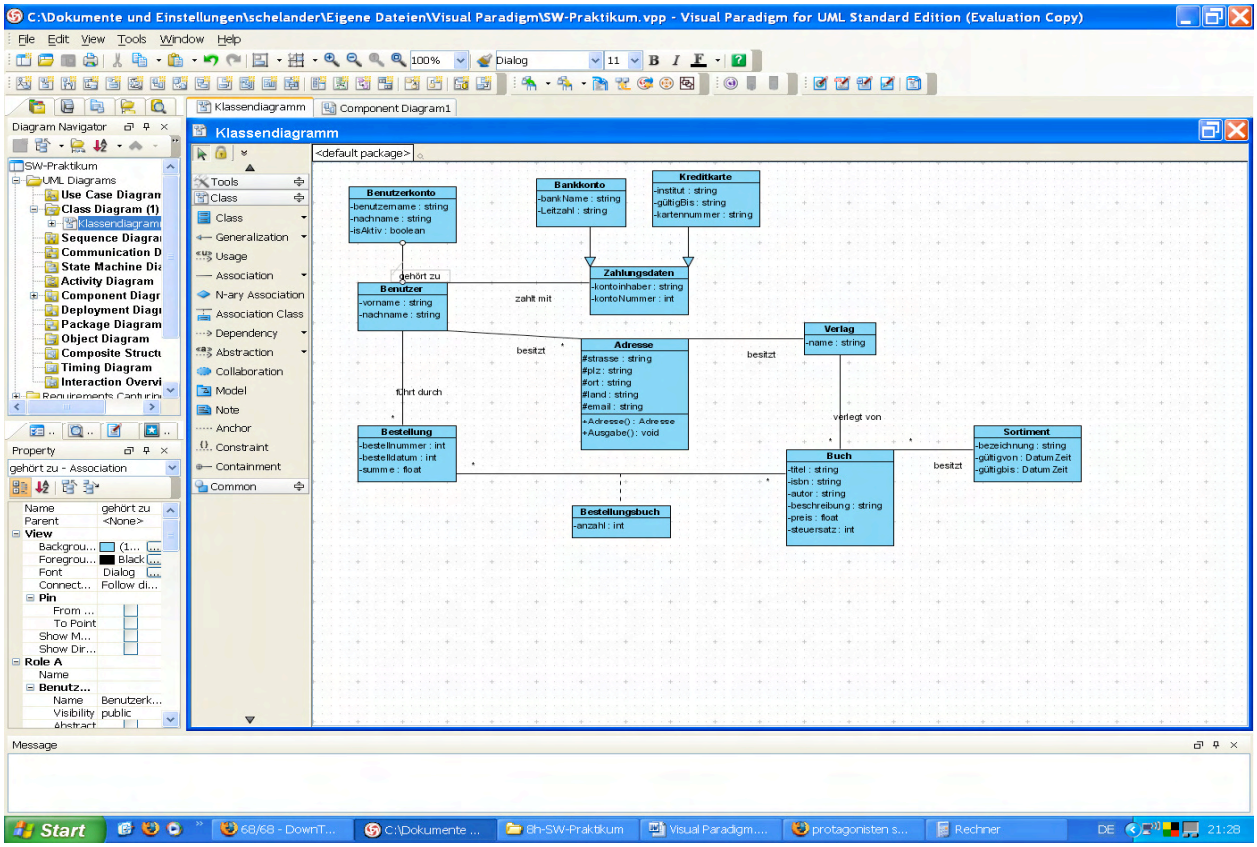
Schritt	Vorgangsweise	Benötigte Aktionen
01 - Neues Projekt anlegen.	LK auf Icon *Create a new project,* Name eingeben, bestätigen. Mit Maus über Menüleiste und *Tools* Projekteigenschaften einstellen.	Maus: 6 Tastatur: 3 Wechsel: 1 10
02 - Anlegen eines neuen Klassendiagramms im Projekt.	LK auf Icon *Create a new UML 2 static class diagram*. LK in Zeichenbereich.	Maus: 2 Tastatur: 0 Wechsel: 0 2
03 - Anlegen der Klassen: Benutzerkonto, Benutzer, Bestellung, Adresse, Zahlungsdaten, Bankkonto, Kreditkarte, Buch, Verlag, Sortiment und Bestellungsbuch.	Links im Wergzeugsfenster per LK auf das Klassensymbol *Create a class* und danach LK auf die Arbeitsfläche. Tastatureingabe der Klassenbezeichnungen. Diese Aktivitäten werden für alle neuen Klassen wiederholt.	Maus: 22 Tastatur: 11 Wechsel: 21 54

Schritt	Vorgangsweise	Benötigte Aktionen
04 - Hinzufügen der Attribute der Klasse Benutzerkonto.	LK auf Icon *Create an attribute,* Eingabe der Attributbezeichnung sowie des Typs. Setzen der Sichtbarkeit	Maus: 27 Tastatur: 3 Wechsel: 4 34
05 - Hinzufügen der Attribute der Klasse Adresse.	Analog zu Schritt 04.	Maus: 45 Tastatur: 5 Wechsel: 14 64
06 - Hinzufügen der Attribute der Klasse Benutzer.	Analog zu Schritt 04.	Maus: 18 Tastatur: 2 Wechsel: 3 23
07 - Hinzufügen der Attribute der Klasse Zahlungsdaten.	Analog zu Schritt 04.	Maus: 18 Tastatur: 2 Wechsel: 5 25
08 - Hinzufügen der Attribute der Klasse Bankkonto.	Analog zu Schritt 04.	Maus: 18 Tastatur: 2 Wechsel: 5 25
09 - Hinzufügen der Attribute der Klasse Kreditkarte.	Analog zu Schritt 04.	Maus: 27 Tastatur: 3 Wechsel: 8 38
10 - Hinzufügen der Attribute der Klasse Buch.	Analog zu Schritt 04.	Maus: 54 Tastatur: 6 Wechsel: 17 77
11 - Hinzufügen der Attribute der Klasse Verlag.	Analog zu Schritt 04.	Maus: 9 Tastatur: 1 Wechsel: 2 12
12 - Hinzufügen der Attribute der Klasse Sortiment.	Analog zu Schritt 04.	Maus: 27 Tastatur: 3 Wechsel: 8 38
13 - Hinzufügen der Attribute der Klasse Bestellungsbuch.	Analog zu Schritt 04.	Maus: 9 Tastatur: 1 Wechsel: 2 6
14 - Anlegen einer Assoziation zwischen den Klassen Benutzerkonto und Benutzer.	LK auf das Icon *Create an assoziation* und Verbinden der beiden Klassen durch Benutzung der Maus. Angabe der Muliplizitäten durch Listenauswahl und Benennung der Assoziation, Navigierbarkeit ausschalten.	Maus: 8 Tastatur: 3 Wechsel: 6 17

Schritt	Vorgangsweise	Benötigte Aktionen
15 - Anlegen einer Assoziation zwischen den Klassen Benutzer und Bestellung.	Analog zu Schritt 14.	Maus: 9 Tastatur: 2 Wechsel: 4 15
16 - Anlegen einer Assoziation zwischen den Klassen Benutzer und Adresse.	Analog zu Schritt 14.	Maus: 9 Tastatur: 2 Wechsel: 4 15
17 - Anlegen einer Assoziation zwischen den Klassen Benutzer und Zahlungsdaten.	Analog zu Schritt 14.	Maus: 8 Tastatur: 3 Wechsel: 6 17
18 - Anlegen einer Assoziation zwischen den Klassen Adresse und Verlag.	Analog zu Schritt 14.	Maus: 8 Tastatur: 3 Wechsel: 6 17
19 – Anlegen einer Assoziation zwischen den beiden Klassen Bestellung und Buch.	Analog zu Schritt 14.	Maus: 11 Tastatur: 1 Wechsel: 2 14
20 - Anlegen einer Assoziation zwischen den Klassen Verlag und Buch.	Analog zu Schritt 14.	Maus: 9 Tastatur: 2 Wechsel: 4 15
21 - Anlegen einer Assoziation zwischen den Klassen Buch und Sortiment.	Analog zu Schritt 14.	Maus: 10 Tastatur: 3 Wechsel: 6 19
22 - Verbinden der Assoziationsklasse Bestellungsbuch und der Assoziation.	LK auf das Icon *Create an association class* und verbinden der Klasse mit der Assoziation.	Maus: 3 Tastatur: 0 Wechsel: 0 3

Schritt	Vorgangsweise	Benötigte Aktionen
23 - Einfügen der beiden Generalisierungen zwischen den Klassen Zahlungsdaten und Bankkonto bzw. Zahlungsdaten und Kreditkarte.	LK auf Icon *Create an inheritance link* und verbinden der jeweiligen Klassen.	Maus: 4 Tastatur: 0 Wechsel: 0
		4
GESAMT		**539**

6.4.6. Klassendiagramm – Visual Paradigm



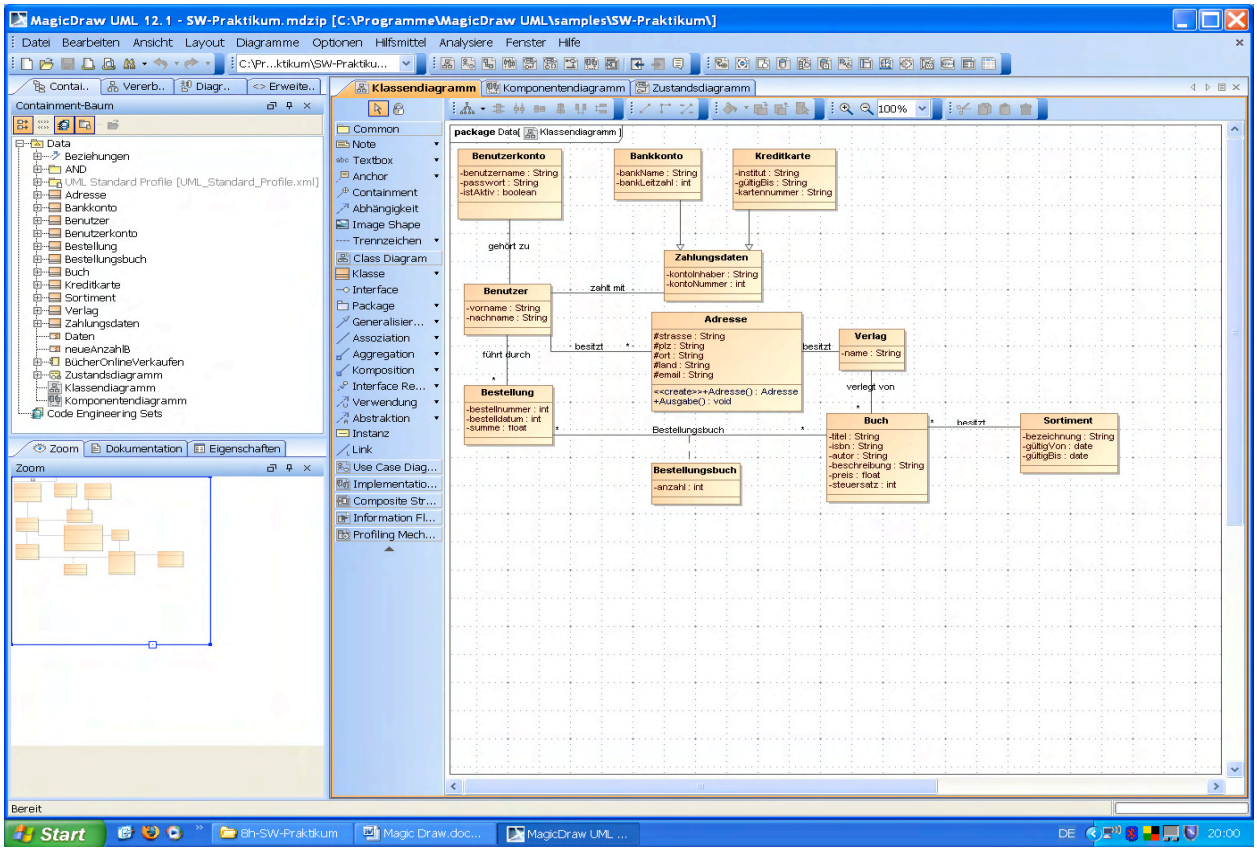
Schritt	Vorgangsweise	Benötigte Aktionen
01 - Neues Projekt anlegen.	LK auf Icon *New Project,* Name eingeben, bestätigen.	Maus: 1 Tastatur: 2 Wechsel: 1
		4
02 - Anlegen eines neuen Klassendiagramms im Projekt.	LK auf Icon *New Class Diagram*.	Maus: 1 Tastatur: 0 Wechsel: 0
		1
03 - Anlegen der Klassen: Benutzerkonto, Benutzer, Bestellung, Adresse, Zahlungsdaten, Bankkonto, Kreditkarte, Buch, Verlag, Sortiment und Bestellungsbuch.	Icon *Class* aus dem Werkzeugbereich in den Zeichenbereich ziehen. DK in den Kopfbereich der Klasse, Bezeichnung eingeben. Für alle Klassen wiederholen.	Maus: 22 Tastatur: 11 Wechsel: 21
		54

Schritt	Vorgangsweise	Benötigte Aktionen
04 - Hinzufügen der Attribute der Klasse Benutzerkonto.	Mit der Maus in die Klasse fahren, LK auf *Open Specification*, Registerkarte *Attributes* auswählen, Attribute mit *Add* hinzufügen, benennen, Sichtbarkeiten und Typen setzen.	Maus: 21 Tastatur: 3 Wechsel: 6 30
05 - Hinzufügen der Attribute der Klasse Adresse.	Analog zu Schritt 04.	Maus: 33 Tastatur: 5 Wechsel: 10 48
06 - Hinzufügen der Attribute der Klasse Benutzer.	Analog zu Schritt 04.	Maus: 15 Tastatur: 2 Wechsel: 4 21
07 - Hinzufügen der Attribute der Klasse Zahlungsdaten.	Analog zu Schritt 04.	Maus: 15 Tastatur: 2 Wechsel: 4 21
08 - Hinzufügen der Attribute der Klasse Bankkonto.	Analog zu Schritt 04.	Maus: 15 Tastatur: 2 Wechsel: 4 21
09 - Hinzufügen der Attribute der Klasse Kreditkarte.	Analog zu Schritt 04.	Maus: 15 Tastatur: 3 Wechsel: 2 21
10 - Hinzufügen der Attribute der Klasse Buch.	Analog zu Schritt 04.	Maus: 39 Tastatur: 6 Wechsel: 12 57
11 - Hinzufügen der Attribute der Klasse Verlag.	Analog zu Schritt 04.	Maus: 9 Tastatur: 1 Wechsel: 2 12
12 - Hinzufügen der Attribute der Klasse Sortiment.	Analog zu Schritt 04.	Maus: 15 Tastatur: 3 Wechsel: 2 21
13 - Hinzufügen der Attribute der Klasse Bestellungsbuch.	Analog zu Schritt 04.	Maus: 9 Tastatur: 1 Wechsel: 2 6
14 - Anlegen einer Assoziation zwischen den Klassen Benutzerkonto und Benutzer.	LK auf *Association*. LK auf die erste zu verbindende Klasse, gedrückt halten, Assoziation zur zweiten Klasse ziehen. Maustaste auslassen.	Maus: 2 Tastatur: 2 Wechsel: 1 5

Schritt	Vorgangsweise	Benötigte Aktionen
15 - Anlegen einer Assoziation zwischen den Klassen Benutzer und Bestellung.	Analog zu Schritt 14. Zusätzlich Angabe einer Multiplizität (RK mit Maus auf Assoziation. Auswahl über *Rolle, Multiplizität* und LK auf die gewünschte Multiplizität.	Maus: 4 Tastatur: 2 Wechsel: 2 8
16 - Anlegen einer Assoziation zwischen den Klassen Benutzer und Adresse.	Analog zu Schritt 14.	Maus: 4 Tastatur: 2 Wechsel: 2 8
17 - Anlegen einer Assoziation zwischen den Klassen Benutzer und Zahlungsdaten.	Analog zu Schritt 14.	Maus: 2 Tastatur: 2 Wechsel: 1 5
18 - Anlegen einer Assoziation zwischen den Klassen Adresse und Verlag.	Analog zu Schritt 14.	Maus: 2 Tastatur: 2 Wechsel: 1 5
19 – Anlegen einer Assoziation zwischen den beiden Klassen Bestellung und Buch.	Analog zu Schritt 14.	Maus: 6 Tastatur: 1 Wechsel: 2 9
20 - Anlegen einer Assoziation zwischen den Klassen Verlag und Buch.	Analog zu Schritt 14.	Maus: 4 Tastatur: 2 Wechsel: 2 8
21 - Anlegen einer Assoziation zwischen den Klassen Buch und Sortiment.	Analog zu Schritt 14.	Maus: 4 Tastatur: 2 Wechsel: 2 8
22 - Verbinden der Assoziationsklasse Bestellungsbuch und der Assoziation.	LK auf das Icon *Association Class*. Anschließend LK auf Assoziation, Maustaste gedrückt halten, zur Assoziationsklasse ziehen, Maustaste auslassen.	Maus: 2 Tastatur: 0 Wechsel: 0 2

Schritt	Vorgangsweise	Benötigte Aktionen
23 - Einfügen der beiden Generalisierungen zwischen den Klassen Zahlungsdaten und Bankkonto bzw. Zahlungsdaten und Kreditkarte.	LK auf Icon *Generalization* und verbinden der jeweiligen Klassen (LK auf eine Klasse, ziehen, Maustaste über der zweiten Klasse auslassen).	Maus: 4 Tastatur: 0 Wechsel: 0 4
GESAMT		**379**

6.4.7. Klassendiagramm - Magic Draw



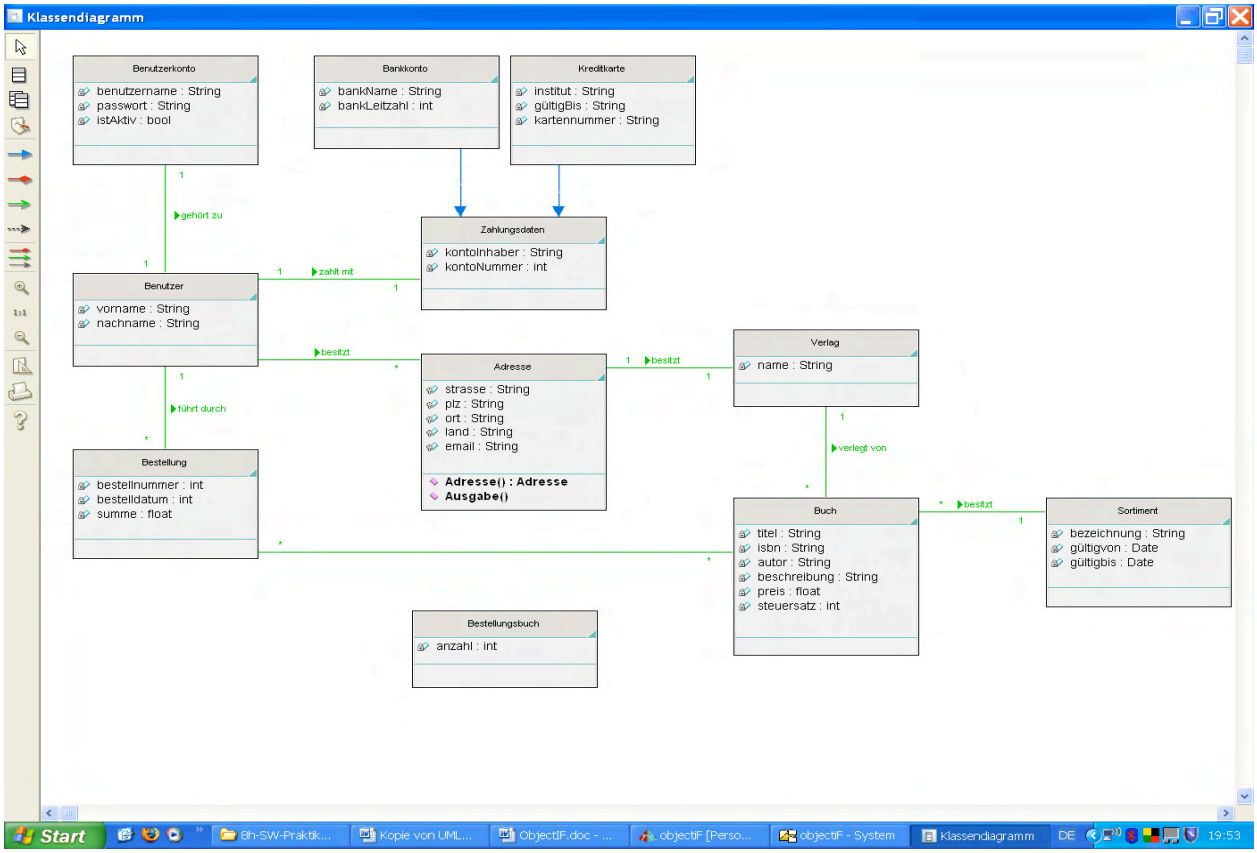
Schritt	Vorgangsweise	Benötigte Aktionen
01 - Neues Projekt anlegen.	LK auf Icon *Neues Projekt,* Name eingeben, bestätigen.	Maus: 1 Tastatur: 2 Wechsel: 1
		4
02 - Anlegen eines neuen Klassendiagramms im Projekt.	LK auf Icon *Klassendiagramm*. Diagramm benennen und bestätigen.	Maus: 1 Tastatur: 2 Wechsel: 1
		4
03 - Anlegen der Klassen: Benutzerkonto, Benutzer, Bestellung, Adresse, Zahlungsdaten, Bankkonto, Kreditkarte, Buch, Verlag, Sortiment und Bestellungsbuch.	LK auf den Button *Klasse* im Werkzeugbereich. Anschließend LK im Zeichenbereich. Bezeichnung eingeben. Für alle Klassen wiederholen.	Maus: 22 Tastatur: 11 Wechsel: 21
		54

Schritt	Vorgangsweise	Benötigte Aktionen
04 - Hinzufügen der Attribute der Klasse Benutzerkonto.	RK auf die Klasse. LK auf Spezifikation. LK auf Attribute. Die einzelnen Attribute über „Anlegen“ hinzufügen und beschriften. Typ bestimmen. Default-Wert für Sichtbarkeit ist „private“.	Maus: 16 Tastatur: 3 Wechsel: 6 25
05 - Hinzufügen der Attribute der Klasse Adresse.	Analog zu Schritt 04.	Maus: 34 Tastatur: 5 Wechsel: 10 49
06 - Hinzufügen der Attribute der Klasse Benutzer.	Analog zu Schritt 04.	Maus: 12 Tastatur: 2 Wechsel: 4 18
07 - Hinzufügen der Attribute der Klasse Zahlungsdaten.	Analog zu Schritt 04.	Maus: 12 Tastatur: 2 Wechsel: 4 18
08 - Hinzufügen der Attribute der Klasse Bankkonto.	Analog zu Schritt 04.	Maus: 12 Tastatur: 2 Wechsel: 4 18
09 - Hinzufügen der Attribute der Klasse Kreditkarte.	Analog zu Schritt 04.	Maus: 16 Tastatur: 3 Wechsel: 6 25
10 - Hinzufügen der Attribute der Klasse Buch.	Analog zu Schritt 04.	Maus: 28 Tastatur: 6 Wechsel: 12 46
11 - Hinzufügen der Attribute der Klasse Verlag.	Analog zu Schritt 04.	Maus: 8 Tastatur: 1 Wechsel: 2 11
12 - Hinzufügen der Attribute der Klasse Sortiment.	Analog zu Schritt 04.	Maus: 16 Tastatur: 3 Wechsel: 6 25
13 - Hinzufügen der Attribute der Klasse Bestellungsbuch.	Analog zu Schritt 04.	Maus: 8 Tastatur: 1 Wechsel: 2 11

Schritt	Vorgangsweise	Benötigte Aktionen
14 - Anlegen einer Assoziation zwischen den Klassen Benutzerkonto und Benutzer.	LK auf *Assoziation*. LK auf die erste zu verbindende Klasse, gedrückt halten, Assoziation zur zweiten Klasse ziehen. Maustaste auslassen. RK auf die Assoziationsenden und Multiplizitäten setzen.	Maus:2 Tastatur:1 Wechsel:1 4
15 - Anlegen einer Assoziation zwischen den Klassen Benutzer und Bestellung.	Analog zu Schritt 14.	Maus: 4 Tastatur: 1 Wechsel: 2 7
16 - Anlegen einer Assoziation zwischen den Klassen Benutzer und Adresse.	Analog zu Schritt 14.	Maus: 4 Tastatur: 1 Wechsel: 2 7
17 - Anlegen einer Assoziation zwischen den Klassen Benutzer und Zahlungsdaten.	Analog zu Schritt 14.	Maus: 2 Tastatur: 1 Wechsel: 1 4
18 - Anlegen einer Assoziation zwischen den Klassen Adresse und Verlag.	Analog zu Schritt 14.	Maus: 2 Tastatur: 1 Wechsel: 1 4
19 – Anlegen einer Assoziation zwischen den beiden Klassen Bestellung und Buch.	Analog zu Schritt 14.	Maus: 6 Tastatur: 0 Wechsel: 0 6
20 - Anlegen einer Assoziation zwischen den Klassen Verlag und Buch.	Analog zu Schritt 14.	Maus: 4 Tastatur: 1 Wechsel: 2 7
21 - Anlegen einer Assoziation zwischen den Klassen Buch und Sortiment.	Analog zu Schritt 14.	Maus: 4 Tastatur: 1 Wechsel: 2 7
22 - Verbinden der Assoziationsklasse Bestellungsbuch und der Assoziation.	LK auf *Assoziationsklasse*. Anschließend LK auf Assoziation, Maustaste gedrückt halten, zur Assoziationsklasse ziehen, Maustaste auslassen.	Maus: 2 Tastatur: 0 Wechsel: 0 2

Schritt	Vorgangsweise	Benötigte Aktionen
23 - Einfügen der beiden Generalisierungen zwischen den Klassen Zahlungsdaten und Bankkonto bzw. Zahlungsdaten und Kreditkarte.	LK auf Icon *Generalisierung* und verbinden der jeweiligen Klassen (LK auf eine Klasse, ziehen, Maustaste über der zweiten Klasse auslassen).	Maus: 4 Tastatur: 0 Wechsel: 0 4
GESAMT		**360**

6.4.8. Klassendiagramm – ObjectIF



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Neues Projekt anlegen.	LK auf *System*. LK auf *Öffnen/Anlegen*. *Ein neues System anlegen* auswählen und weiter dem Assistenten folgen.	Maus: 8 Tastatur: 2 Wechsel: 2 12
02 - Anlegen eines neuen Klassendiagramms im Projekt.	LK auf Icon *Design*. RK in den leeren Bereich. Maus über *Anlegen* und LK auf *Klassendiagramm*. Name vergeben und bestätigen.	Maus: 3 Tastatur: 2 Wechsel: 1 6
03 - Anlegen der Klassen: Benutzerkonto, Benutzer, Bestellung, Adresse, Zahlungsdaten, Bankkonto, Kreditkarte, Buch, Verlag, Sortiment und Bestellungsbuch.	LK auf Icon *Klasse anlegen* und LK im Zeichenbereich. Name vergeben und bestätigen. Für alle Klassen wiederholen.	Maus: 22 Tastatur: 22 Wechsel: 21 65

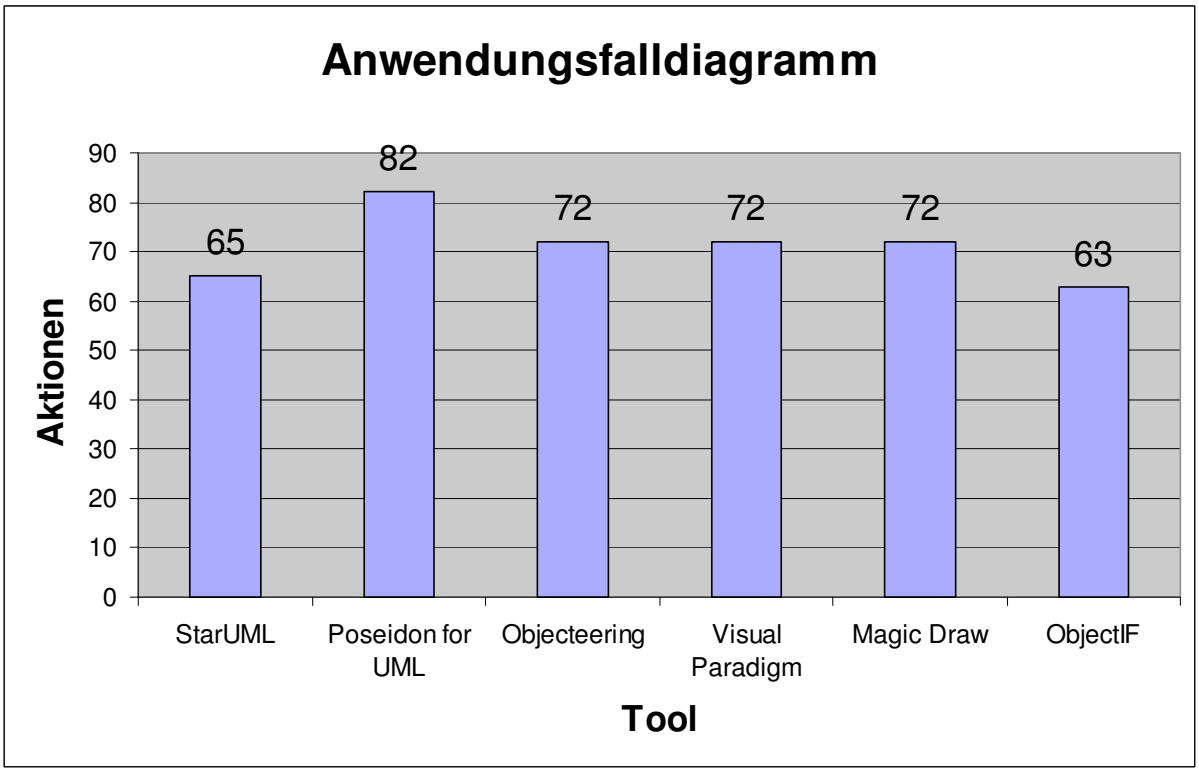
Schritt	Vorgangsweise	Benötigte Aktionen
04 - Hinzufügen der Attribute der Klasse Benutzerkonto.	RK auf Klasse. Maus über Anlegen, LK auf *Attribut*. Name vergeben, Sichtbarkeiten und Typ bestimmen. Nach Eingabe des letzten Attributs LK auf *Schließen*. Default-Sichtabarkeit ist „private“.	Maus: 15 Tastatur: 3 Wechsel: 6 24
05 - Hinzufügen der Attribute der Klasse Adresse.	Analog zu Schritt 04. Zusätzlich Sichtbarkeit ändern.	Maus: 33 Tastatur: 5 Wechsel: 10 48
06 - Hinzufügen der Attribute der Klasse Benutzer.	Analog zu Schritt 04.	Maus: 11 Tastatur: 2 Wechsel: 4 17
07 - Hinzufügen der Attribute der Klasse Zahlungsdaten.	Analog zu Schritt 04.	Maus: 11 Tastatur: 2 Wechsel: 4 17
08 - Hinzufügen der Attribute der Klasse Bankkonto.	Analog zu Schritt 04.	Maus: 11 Tastatur: 2 Wechsel: 4 17
09 - Hinzufügen der Attribute der Klasse Kreditkarte.	Analog zu Schritt 04.	Maus: 15 Tastatur: 3 Wechsel: 6 24
10 - Hinzufügen der Attribute der Klasse Buch.	Analog zu Schritt 04.	Maus: 27 Tastatur: 6 Wechsel: 12 45
11 - Hinzufügen der Attribute der Klasse Verlag.	Analog zu Schritt 04.	Maus: 7 Tastatur: 1 Wechsel: 2 10
12 - Hinzufügen der Attribute der Klasse Sortiment.	Analog zu Schritt 04.	Maus: 15 Tastatur: 3 Wechsel: 6 24
13 - Hinzufügen der Attribute der Klasse Bestellungsbuch.	Analog zu Schritt 04.	Maus: 7 Tastatur: 1 Wechsel: 2 10

Schritt	Vorgangsweise	Benötigte Aktionen
14 - Anlegen einer Assoziation zwischen den Klassen Benutzerkonto und Benutzer.	LK auf *Assoziation*. LK auf die erste zu verbindende Klasse, LK auf die zweite Klasse. LK auf Mauszeigersymbol. LK auf Assoziation. LK in das Bezeichnungsfeld, Name eingeben. Rollen spezifizieren.	Maus: 11 Tastatur: 1 Wechsel: 2 14
15 - Anlegen einer Assoziation zwischen den Klassen Benutzer und Bestellung.	Analog zu Schritt 14. Hier stimmen die Default-Werte der Rollen.	Maus: 7 Tastatur: 2 Wechsel: 1 10
16 - Anlegen einer Assoziation zwischen den Klassen Benutzer und Adresse.	Analog zu Schritt 14.	Maus: 7 Tastatur: 2 Wechsel: 1 10
17 - Anlegen einer Assoziation zwischen den Klassen Benutzer und Zahlungsdaten.	Analog zu Schritt 14.	Maus:11 Tastatur: 1 Wechsel: 2 14
18 - Anlegen einer Assoziation zwischen den Klassen Adresse und Verlag.	Analog zu Schritt 14.	Maus: 11 Tastatur: 1 Wechsel: 2 14
19 – Anlegen einer Assoziation zwischen den beiden Klassen Bestellung und Buch.	Analog zu Schritt 14.	Maus: 9 Tastatur: 0 Wechsel: 0 9
20 - Anlegen einer Assoziation zwischen den Klassen Verlag und Buch.	Analog zu Schritt 14.	Maus: 7 Tastatur: 2 Wechsel: 1 10
21 - Anlegen einer Assoziation zwischen den Klassen Buch und Sortiment.	Analog zu Schritt 14.	Maus: 11 Tastatur: 1 Wechsel: 2 14
22 - Verbinden der Assoziationsklasse Bestellungsbuch und der Assoziation.	Keine Assoziationsklasse möglich.	Maus: 0 Tastatur: 0 Wechsel: 0 0

Schritt	Vorgangsweise	Benötigte Aktionen
23 - Einfügen der beiden Generalisierungen zwischen den Klassen Zahlungsdaten und Bankkonto bzw. Zahlungsdaten und Kreditkarte.	LK auf Icon *Generalisierung* und verbinden der jeweiligen Klassen (LK auf eine Klasse, ziehen, Maustaste über der zweiten Klasse auslassen).	Maus: 6 Tastatur: 0 Wechsel: 0 6
GESAMT		**420**

6.5 Erstellung des Anwendungsfalldiagramms

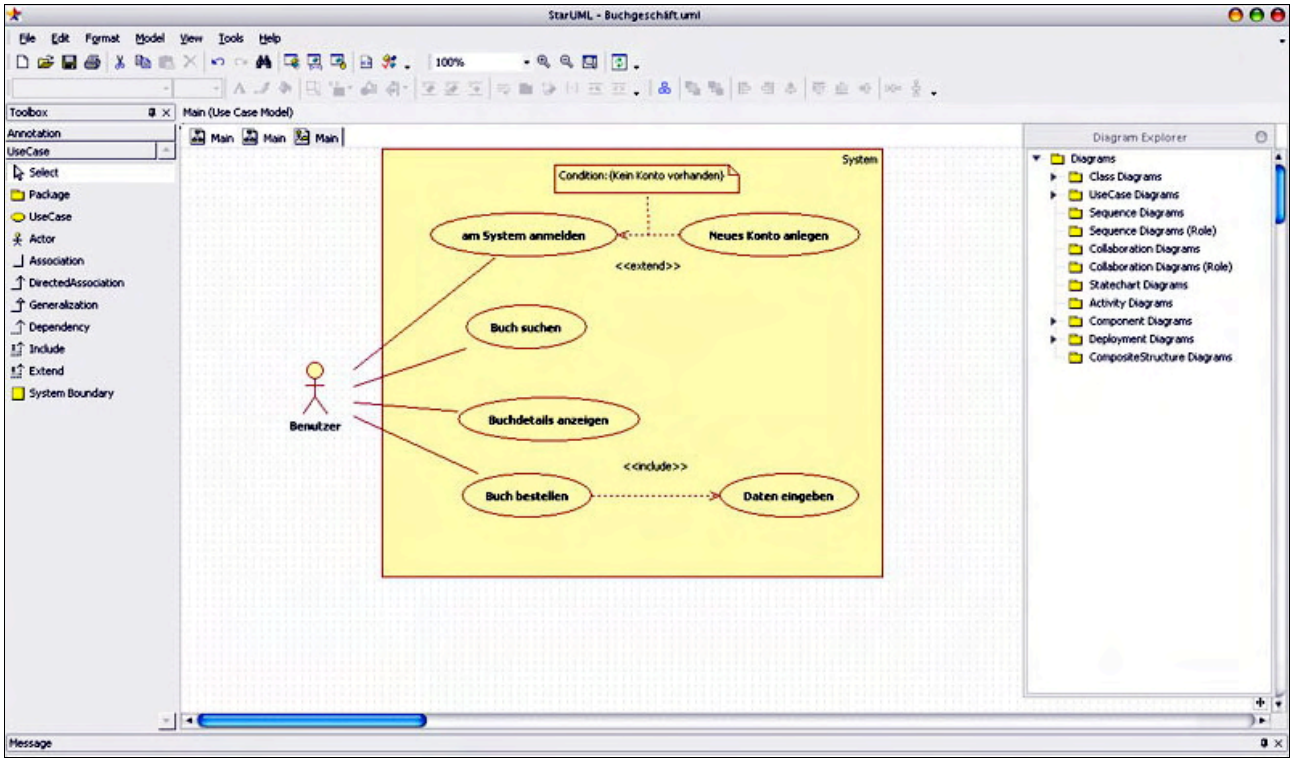
6.5.1. Zusammenfassung der Ergebnisse



6.5.2. Interpretation der Ergebnisse

Im Gegensatz zu den anderen Diagrammarten, unterscheidet sich die Bedienung bei diesem Diagramm zwischen den sechs analysierten Werkzeugen nur geringfügig. Auch die Anzahl der benötigten Aktionen hält sich die Waage. Zu erwähnen ist, dass ObjectIF im Vergleich zu den anderen Tools im Funktionsumfang eingeschränkt ist. Es gibt in ObjectIF nicht die Möglichkeit eine Systemgrenze für Anwendungsfälle einzuzeichnen. Diese Tatsache schlägt sich auch in der geringeren Anzahl an Interaktionen nieder.

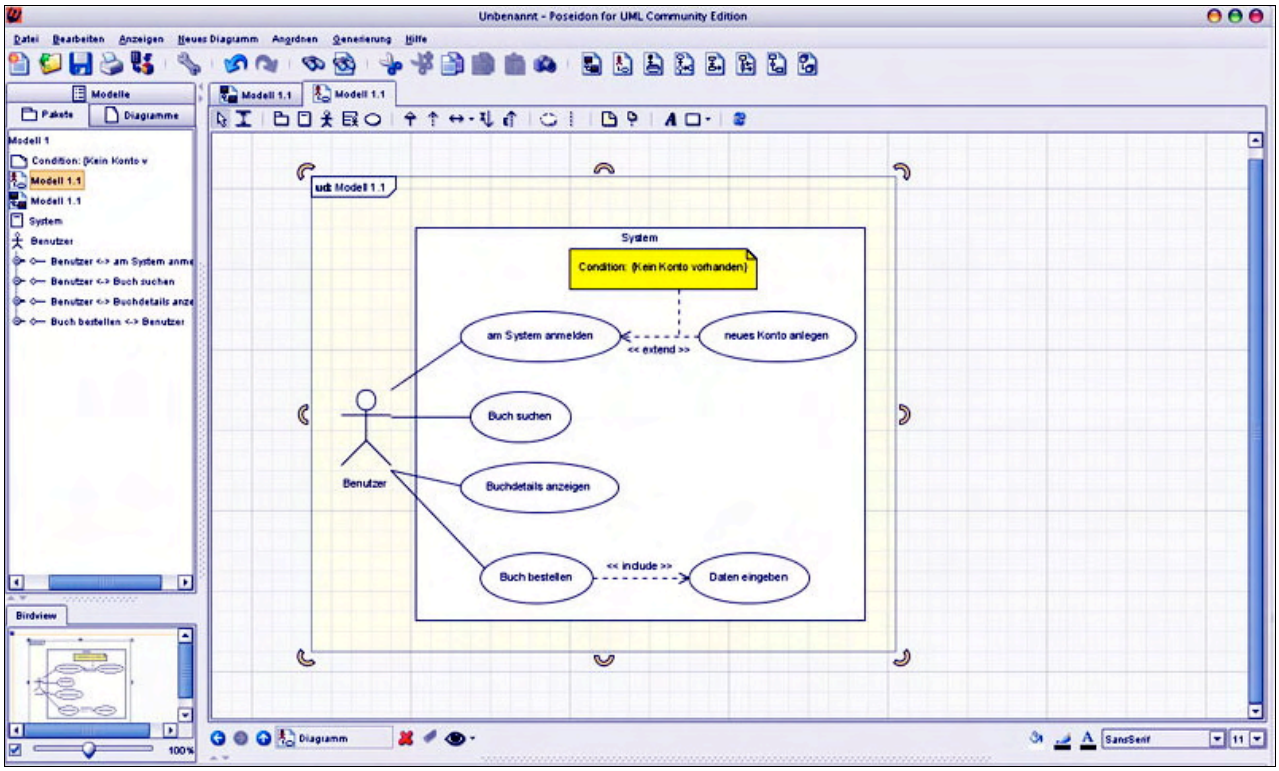
6.5.3. Anwendungsfalldiagramm – StarUML



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Anlegen eines UseCase Diagramms.	DK im Modellexplorer auf *UseCase Diagramm* und DK auf das bereits automatisch erstellte Diagramm.	Maus: 4 Tastatur: 0 Wechsel: 0
		4
02 - Anlegen des Akteurs Kunde.	Auswahl des Werkzeugs *Actor*, LK in den Zeichenbereich und Eingabe des Namens für den neuen Akteur.	Maus: 2 Tastatur: 1 Wechsel: 2
		5
03 - Anlegen der vier Anwendungsfälle.	Auswahl des Werkzeugs *UseCase*, LK in den Zeichenbereich und Eingabe der Bezeichnung des UseCase.	Maus: 8 Tastatur: 4 Wechsel: 8
		20
04 - Anlegen der Beziehungen.	Auswahl durch LK auf dafür vorgesehenes Werkzeug und ziehen der Beziehungen per Maus.	Maus: 8 Tastatur: 0 Wechsel: 0
		8
05 - Hinzufügen der Anwendungsfälle „Neues Konto anlegen“ und „Daten eingeben“.	LK auf *UseCase*-Werkzeug und LK auf Zeichenebene, sowie benennen der beiden Anwendungsfälle.	Maus: 4 Tastatur: 2 Wechsel: 4
		10

Schritt	Vorgangsweise	Benötigte Aktionen
06 - Setzen der include-Beziehung.	Include Werkzeug wählen und Ziehen der Assoziation.	Maus: 2 Tastatur: 0 Wechsel: 0
		2
07 - Setzen der extend-Beziehung.	Extend Werkzeug wählen und Ziehen der Assoziation.	Maus: 2 Tastatur: 0 Wechsel: 0
		2
08 - Hinzufügen der Anmerkung.	Wechseln zur *Annotation*-Werkzeugleiste durch LK, auswählen der Notiz und hinzufügen zum Diagramm durch LK. Eingabe der Notiz.	Maus: 3 Tastatur: 1 Wechsel: 2
		6
09 - Hinzufügen der Systemgrenzen.	Klick auf *System Boundary* in der Werkzeugleiste und Aufziehen des Systems, sowie beschriften	Maus: 2 Tastatur: 2 Wechsel: 4
		8
GESAMT		**65**

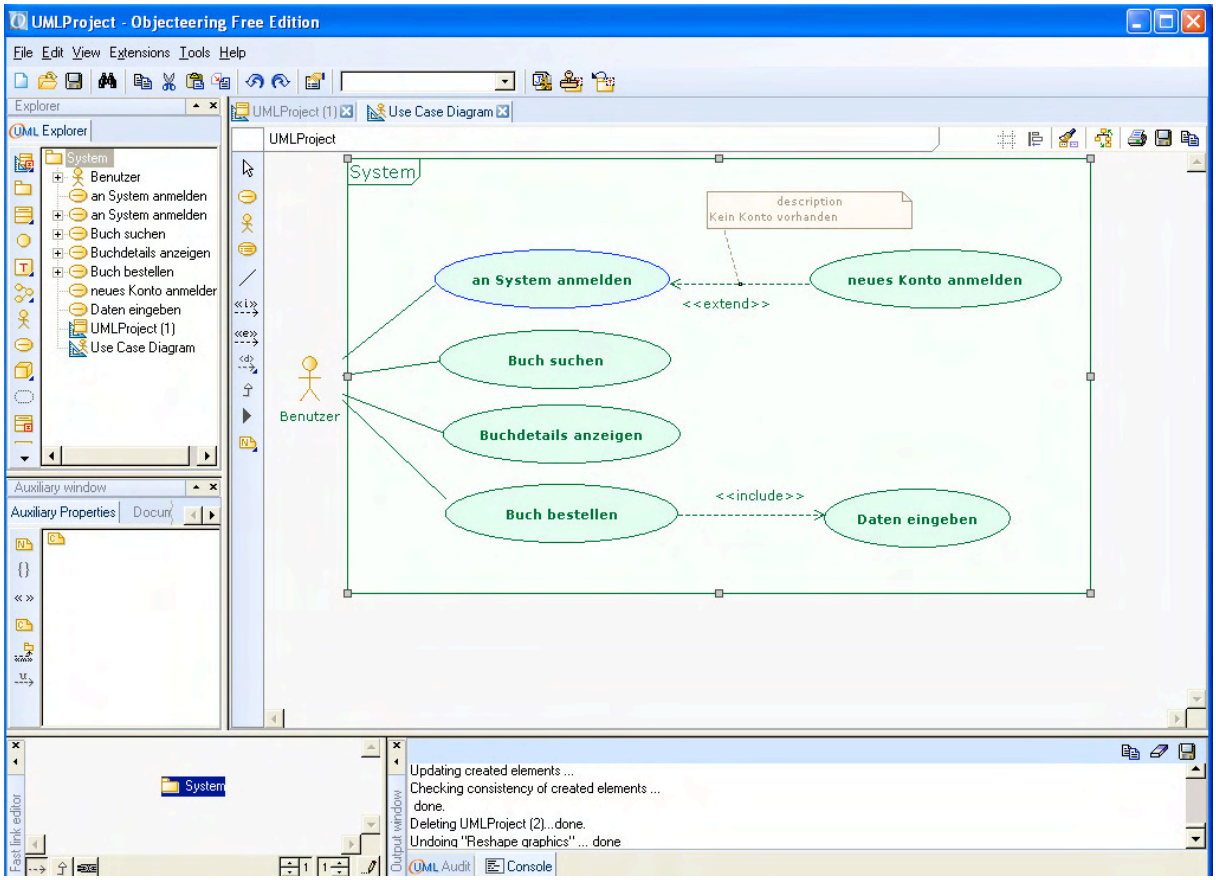
6.5.4. Anwendungsfalldiagramm – Poseidon for UML



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Anlegen eines Anwendungsfalldiagramms.	LK in der Menüleiste auf *Anwendungsfalldiagramm* und benennen des Diagramms.	Maus: 2 Tastatur: 1 Wechsel: 2 5
02 - Anlegen des Akteurs Kunde.	Auswahl des Werkzeugs Actor, LK in den Zeichenbereich und Eingabe des Namens für den neuen Akteur.	Maus: 2 Tastatur: 1 Wechsel: 3 5
03 - Anlegen der vier Anwendungsfälle.	Auswahl des Werkzeugs *UseCase*, LK in den Zeichenbereich und Eingabe der Bezeichnung des UseCase.	Maus: 8 Tastatur: 8 Wechsel: 16 32
04 - Anlegen der Beziehungen.	Auswahl durch LK auf dafür vorgesehenes Werkzeug und ziehen der Beziehungen per Maus.	Maus: 8 Tastatur: 0 Wechsel: 0 8
05 - Hinzufügen der Anwendungsfälle „Neues Konto anlegen“ und „Daten eingeben“.	LK auf *UseCase*-Werkzeug und LK auf Zeichenebene, sowie benennen der beiden Anwendungsfälle.	Maus: 4 Tastatur: 4 Wechsel: 8 16

Schritt	Vorgangsweise	Benötigte Aktionen
06 - Setzen der include-Beziehung.	Include Werkzeug wählen und Ziehen der Assoziation.	Maus: 2 Tastatur: 0 Wechsel: 0
		2
07 - Setzen der extend-Beziehung.	Extend Werkzeug wählen und Ziehen der Assoziation.	Maus: 2 Tastatur: 0 Wechsel: 0
		2
08 - Hinzufügen der Anmerkung.	Durch LK, auswählen der Notiz und hinzufügen zum Diagramm. Eingabe der Notiz.	Maus: 3 Tastatur: 1 Wechsel: 2
		6
09 - Hinzufügen der Systemgrenzen.	Klick auf *Teilsystem* in der Werkzeugleiste und Aufziehen des Systems und Beschriften.	Maus: 2 Tastatur: 2 Wechsel: 4
		8
GESAMT		**82**

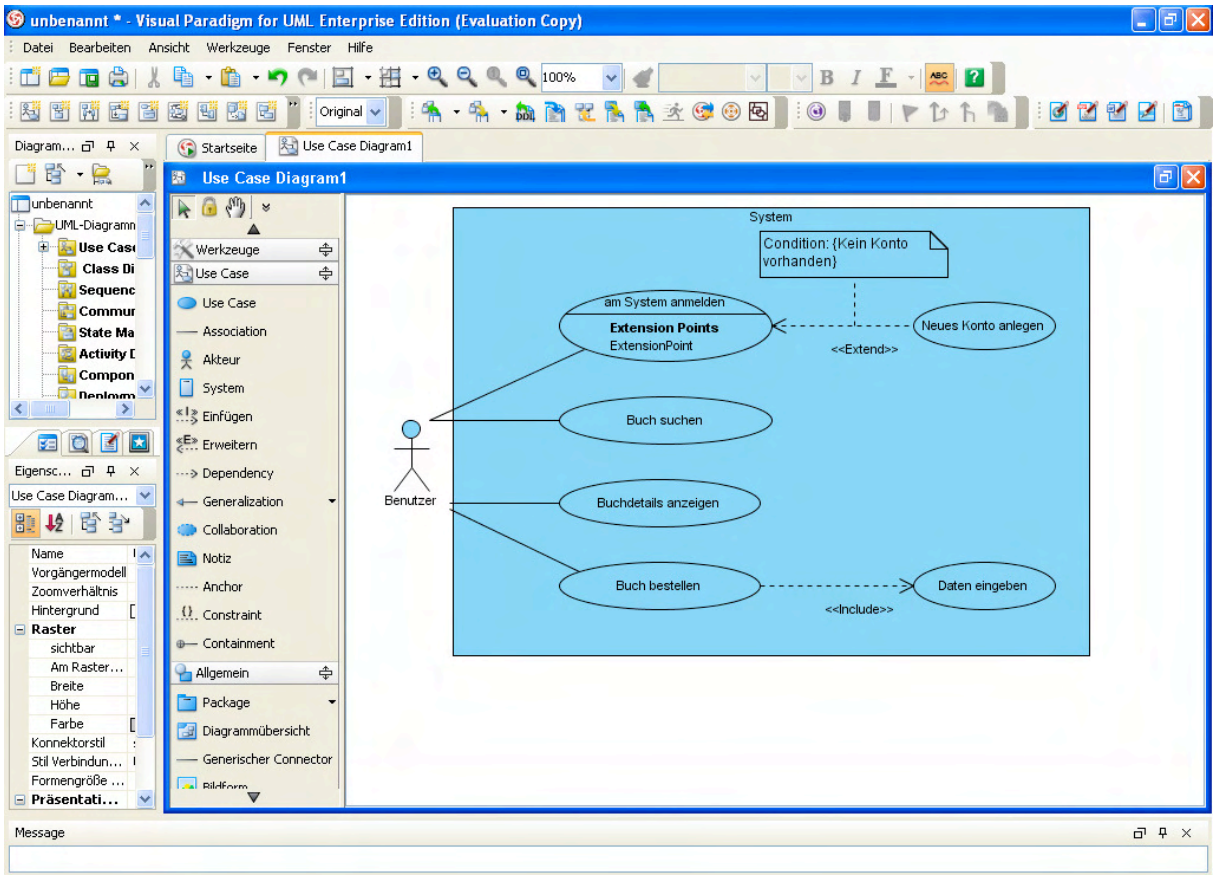
6.5.5. Anwendungsfalldiagramm – Objecteering



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Anlegen eines Anwendungsfall-diagramms.	LK im *UML Explorer* auf *Create a use case diagram* und anschließend 2mal LK auf das UML Project um diese zu benennen.	Maus: 3 Tastatur: 1 Wechsel: 2
		6
02 - Anlegen des Akteurs Kunde.	LK auf C*reate an actor* im Diagram-editor und LK auf die gewünschte Position im Zeichenbereich sowie Eingabe des Namens für den neuen Akteur.	Maus: 2 Tastatur: 1 Wechsel: 2
		5
03 - Anlegen der vier Anwendungsfälle.	LK auf C*reate a cse case*, LK in den Zeichenbereich und Eingabe der Bezeichnung des UseCase.	Maus: 8 Tastatur: 4 Wechsel: 8
		20
04 - Anlegen der Beziehungen.	LK auf *Create an association*, LK auf den Akteur und LK auf den entsprechenden Anwendungsfall.	Maus: 12 Tastatur: 0 Wechsel: 0
		12

Schritt	Vorgangsweise	Benötigte Aktionen
05 - Hinzufügen der Anwendungsfälle „Neues Konto anlegen“ und „Daten eingeben“.	Analog zu Schritt 03.	Maus: 4 Tastatur: 2 Wechsel: 4 10
06 - Setzen der include-Beziehung.	LK auf *Create an inclusion relationship*, LK auf die beiden Anwendungsfälle.	Maus: 3 Tastatur: 0 Wechsel: 0 3
07 - Setzen der extend-Beziehung.	LK auf *Create an extension relationship*, LK auf die beiden Anwendungsfälle.	Maus: 3 Tastatur: 0 Wechsel: 0 3
08 - Hinzufügen der Anmerkung.	LK auf *Create a note*, LK auf die Beziehung und LK auf die gewünschte Position im Zeichenbereich. DK auf die Notiz und Eingabe der Notiz.	Maus: 5 Tastatur: 1 Wechsel: 2 8
09 - Beschriftung der Systemgrenzen.	LK auf den Systemnamen und Eingabe des neuen Namens.	Maus: 2 Tastatur: 1 Wechsel: 2 5
GESAMT		**72**

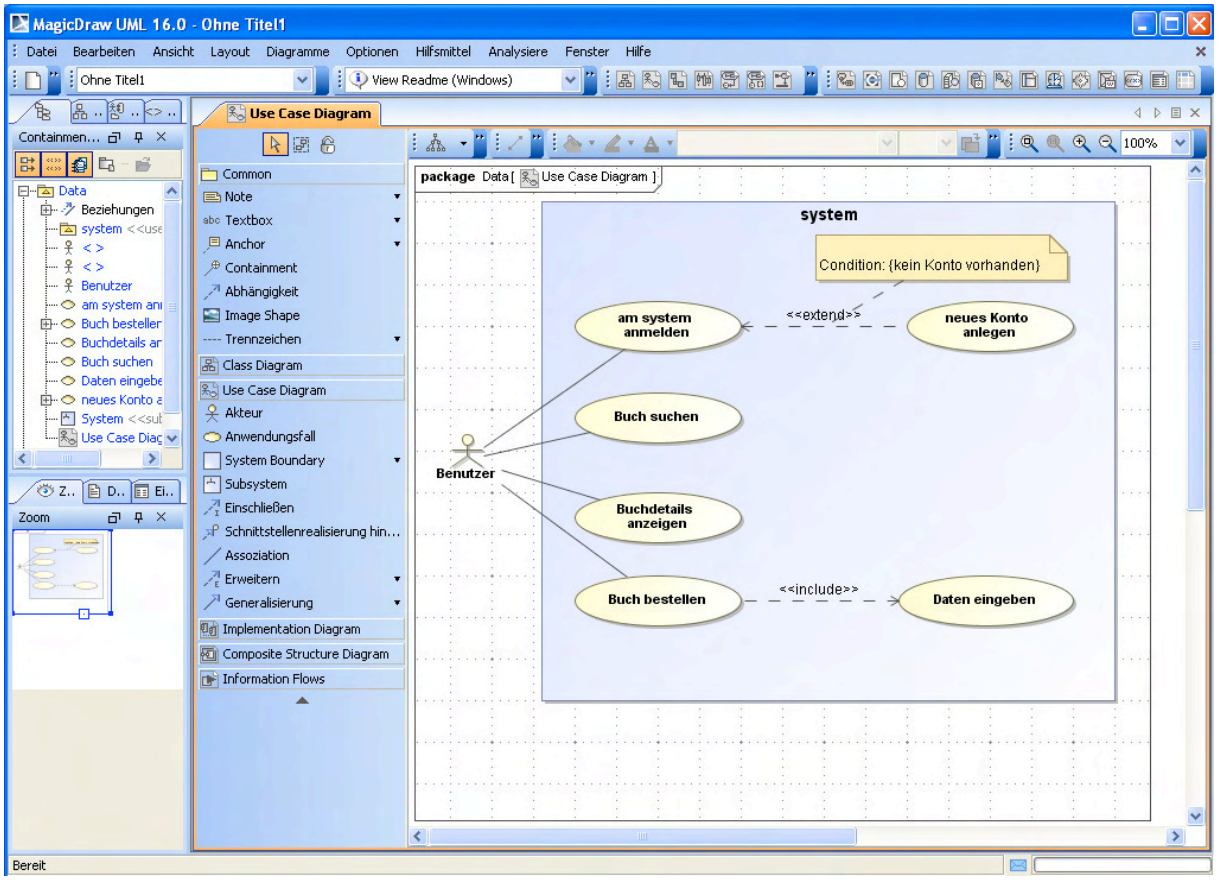
6.5.6. Anwendungsfalldiagramm – Visual Paradigm



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Anlegen eines Anwendungsfalldiagramms.	LK auf *Neues Use Case Diagram* und Eingabe des Diagrammnamen.	Maus: 1 Tastatur: 1 Wechsel: 2
		4
02 - Anlegen des Akteurs Kunde.	Im Werkzeugbereich LK auf *Akteur*, ziehen in den Zeichenbereich und Eingabe der Bezeichnung.	Maus: 2 Tastatur: 1 Wechsel: 2
		5
03 - Anlegen der vier Anwendungsfälle.	Im Werkzeugbereich LK auf Use Case, ziehen in den Zeichenbereich und Eingabe der Bezeichnung.	Maus: 8 Tastatur: 4 Wechsel: 8
		20
04 - Anlegen der Beziehungen.	Auswahl durch LK auf dafür vorgesehenes Werkzeug, LK auf Akteur und ziehen der Beziehungen per Maus.	Maus: 12 Tastatur: 0 Wechsel: 0
		12
05 - Hinzufügen der Anwendungsfälle „Neues Konto anlegen“ und „Daten eingeben“.	Analog zu Schritt 03.	Maus: 4 Tastatur: 2 Wechsel: 4
		10

Schritt	Vorgangsweise	Benötigte Aktionen
06 - Setzen der include-Beziehung.	LK auf *Einfügen*, LK auf Anwendungsfall und ziehen der Beziehungen per Maus.	Maus: 3 Tastatur: 0 Wechsel: 0 3
07 - Setzen der extend-Beziehung.	LK auf *Erweitern*, LK auf Anwendungsfall und ziehen der Beziehungen per Maus.	Maus: 3 Tastatur: 0 Wechsel: 0 3
08 - Hinzufügen der Anmerkung.	LK auf *Notiz* und ziehen in den Zeichenbereich und Eingabe der Beschreibung. LK auf *Anchor*, LK auf die Notiz und ziehen zur Beziehung.	Maus: 5 Tastatur: 1 Wechsel: 2 8
09 - Hinzufügen der Systemgrenzen.	LK auf *System* und ziehen in den Arbeitsbereich. DK auf den Namen und Eingabe der Bezeichnung.	Maus: 4 Tastatur: 1 Wechsel: 2 7
GESAMT		**72**

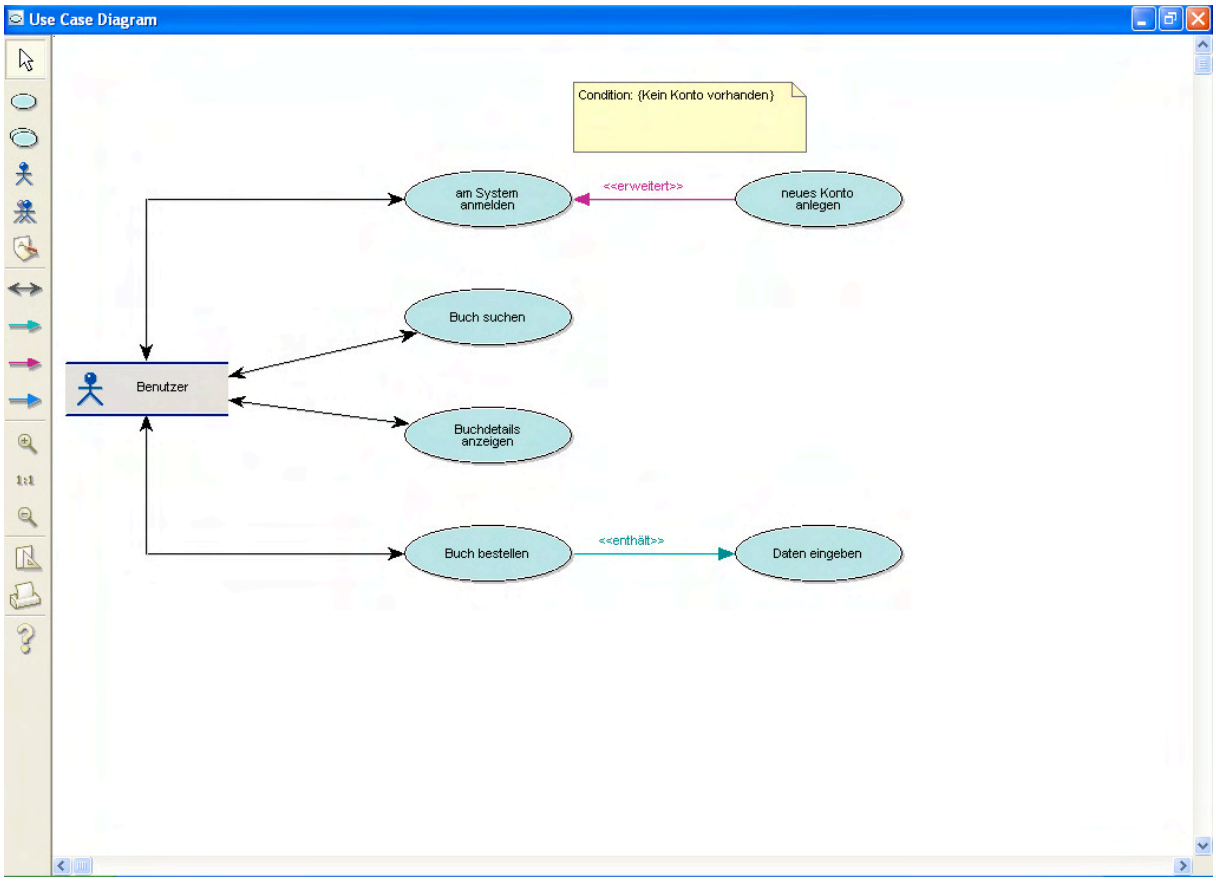
6.5.7. Anwendungsfalldiagramm – Magic Draw



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Anlegen eines Anwendungsfalldiagramms.	LK auf Icon Anwendungsfalldiagramme. Diagramm benennen und bestätigen.	Maus: 1 Tastatur: 2 Wechsel: 2
		5
02 - Anlegen des Akteurs Kunde.	Im Werkzeugbereich LK auf *Akteur*, LK in den Zeichenbereich und Eingabe der Bezeichnung.	Maus: 3 Tastatur: 1 Wechsel: 2
		6
03 - Anlegen der vier Anwendungsfälle.	Im Werkzeugbereich LK auf Anwendungsfall, LK in den Zeichenbereich und Eingabe der Bezeichnung.	Maus: 8 Tastatur: 4 Wechsel: 8
		20
04 - Anlegen der Beziehungen.	Auswahl durch LK auf dafür vorgesehenes Werkzeug, LK auf Akteur und LK auf den Anwendungsfall.	Maus: 12 Tastatur: 0 Wechsel: 0
		12
05 - Hinzufügen der Anwendungsfälle „Neues Konto anlegen“ und „Daten eingeben“.	Analog zu Schritt 03.	Maus: 4 Tastatur: 2 Wechsel: 4
		10

Schritt	Vorgangsweise	Benötigte Aktionen
06 - Setzen der include-Beziehung.	LK auf *Einfügen*, LK auf die beiden Anwendungsfälle.	Maus: 3 Tastatur: 0 Wechsel: 0
		3
07 - Setzen der extend-Beziehung.	LK auf *Erweitern*, LK auf die beiden Anwendungsfälle.	Maus: 3 Tastatur: 0 Wechsel: 0
		3
08 - Hinzufügen der Anmerkung.	LK auf die Beziehung, LK auf *Anchor*, LK auf die gewünschte Position und Texteingabe.	Maus: 3 Tastatur: 1 Wechsel: 2
		6
09 - Hinzufügen der Systemgrenzen.	LK auf *System Boundary,* LK auf die gewünschte Position, DK und Eingabe der Systembezeichnung.	Maus: 4 Tastatur: 1 Wechsel: 2
		7
GESAMT		**72**

6.5.8. Anwendungsfalldiagramm – ObjectIF

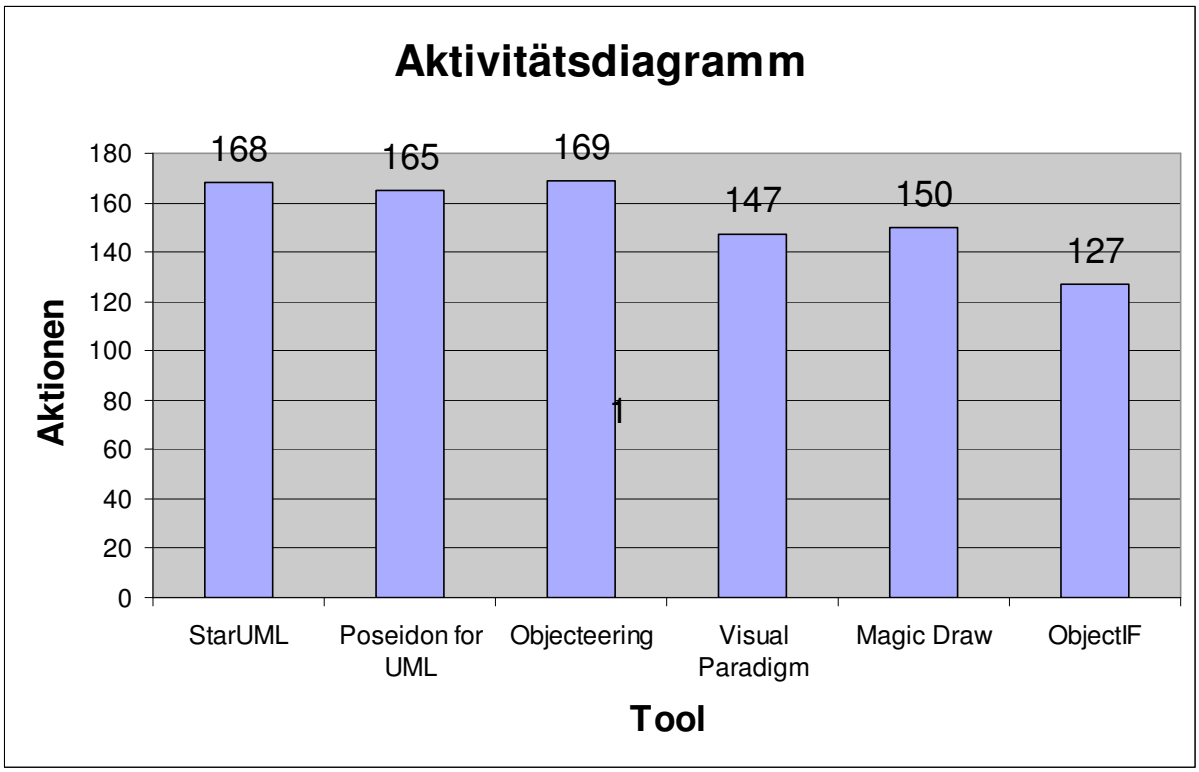


Schritt	Vorgangsweise	Benötigte Aktionen
01 - Anlegen eines Anwendungsfalldiagramms.	LK auf Icon Design. RK in den leeren Bereich. Maus über *Anlegen* und LK auf *Anwendungsfalldiagramm*. Name vergeben und bestätigen.	Maus: 3 Tastatur: 2 Wechsel: 2 7
02 - Anlegen des Akteurs Kunde.	Im Werkzeugbereich LK auf *Akteur anlegen*, LK in den Zeichenbereich und Eingabe der Bezeichnung.	Maus: 2 Tastatur: 1 Wechsel: 2 5
03 - Anlegen der vier Anwendungsfälle.	Im Werkzeugbereich LK auf *Anwendungsfall anlegen*, LK in den Zeichenbereich und Eingabe der Bezeichnung.	Maus: 5 Tastatur: 4 Wechsel: 8 17
04 - Anlegen der Beziehungen.	Auswahl durch LK auf dafür vorgesehenes Werkzeug, LK auf Akteur und LK auf den Anwendungsfall.	Maus: 12 Tastatur: 0 Wechsel: 0 12

Schritt	Vorgangsweise	Benötigte Aktionen
05 - Hinzufügen der Anwendungsfälle „Neues Konto anlegen“ und „Daten eingeben“.	Analog zu Schritt 03.	Maus: 3 Tastatur: 2 Wechsel: 4 9
06 - Setzen der include-Beziehung.	LK auf *Enthält*, LK auf den ersten Anwendungsfall und ziehen der Beziehung.	Maus: 3 Tastatur: 0 Wechsel: 0 3
07 - Setzen der extend-Beziehung.	LK auf *Erweitert*, LK auf den ersten Anwendungsfall und ziehen der Beziehung.	Maus: 3 Tastatur: 0 Wechsel: 0 3
08 - Hinzufügen der Anmerkung.	LK auf *Notiz anlegen*, LK auf die gewünschte Position, DK und Texteingabe.	Maus: 4 Tastatur: 1 Wechsel: 2 7
GESAMT		**63**

6.6 Erstellung des Aktivitätsdiagramms

6.6.1. Zusammenfassung der Ergebnisse

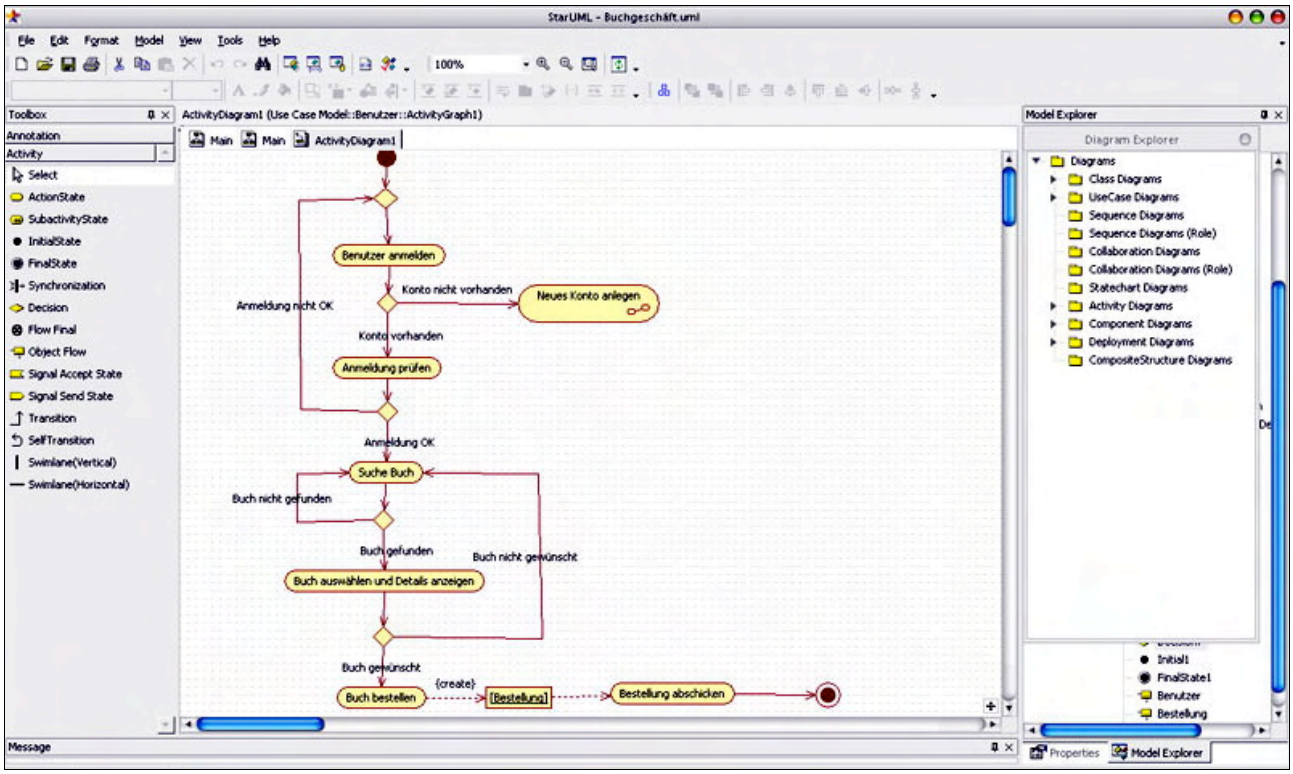


6.6.2. Interpretation der Ergebnisse

Auch hier sind die getesteten Werkzeuge fast gleich auf. Jedoch ist hier anzuführen, dass das Hinzufügen der Beizeichnungen der Kanten bei Poseidon for UML etwas Übung bedarf. Es reicht hier nicht aus nur auf die gewünschte Kante zu klicken um eine Bezeichnung hinzuzufügen, es muss zuerst die Kante per Doppelklick markiert werden und mit einem weiteren Klick auf das Feld <name>, kann erst die Bezeichnung eingegeben werden. Diesen Nachteil macht Poseidon for UML aber in der restlichen Bedienung wieder weg.

ObjectIF verfügt über die Funktion, dass ausgewählte Modellelemente gehalten werden. Dies bedeutet, dass Modellelemente nach einmaliger Auswahl mehrfach angelegt werden können, ohne das Element erneut in der Werkzeugliste anzuklicken. Daraus resultier die geringere Anzahl an Aktionen im Vergleich zu den anderen Tools. Berücksicht man jedoch die Tatsache, dass man mit ObjectIF Notizen nicht sauber darstellen kann und keine Bezeichnung für Endknoten eingeben kann, würden wir Visual Paradigm für die Erstellung dieses Diagrammtyps den Vorzug geben. Mit Visual Paradigm kann man alle Modellierungsaspekte mit einer geringen Anzahl an Aktionen abbilden. Außerdem ist Visual Paradigm leicht und intuitiv zu bedienen.

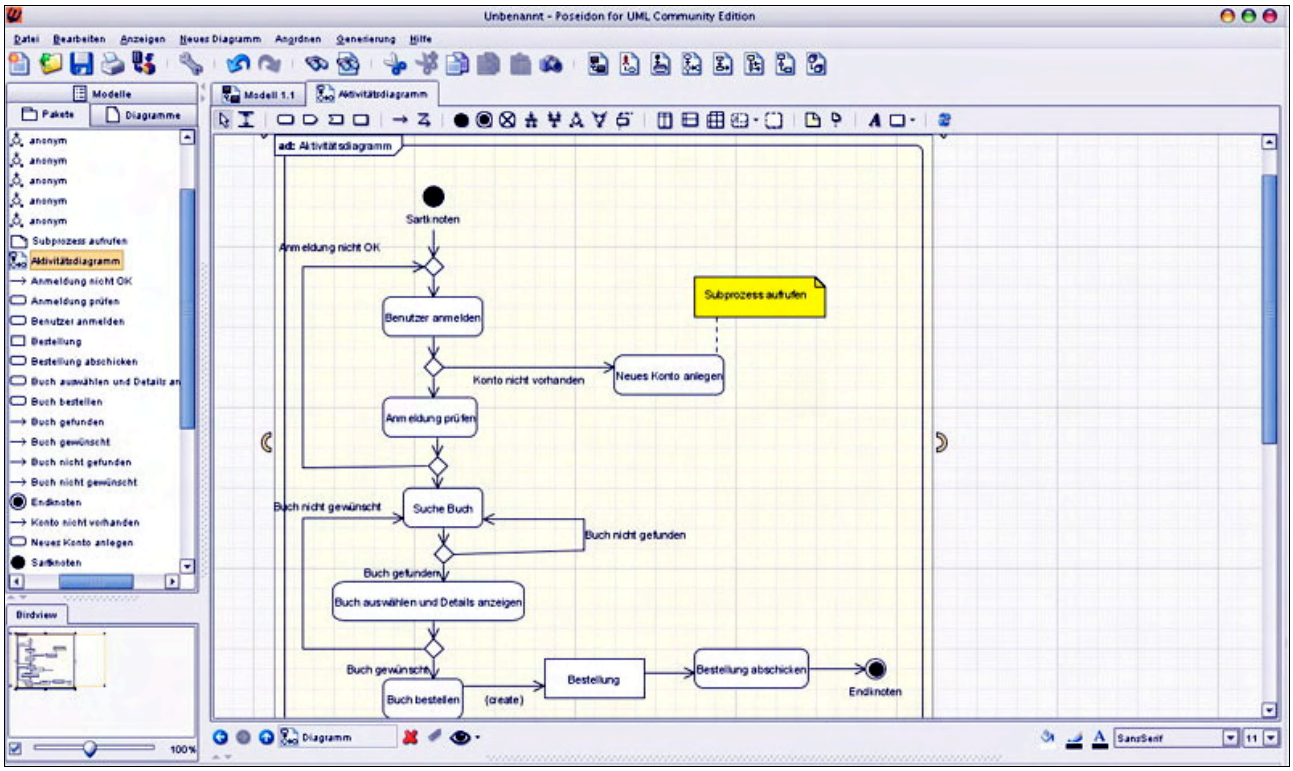
6.6.3. Aktivitätsdiagramm – StarUML



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Neues Aktivitäts-diagramm anlegen.	Im Modellexplorer LK auf *Aktivitäts-diagram*.	Maus: 2 Tastatur: 0 Wechsel: 0
		2
02 - Einfügen des Initi-alknotens.	LK auf *InitialState* und LK in Zeichenebene.	Maus: 2 Tastatur: 0 Wechsel: 0
		2
03 - Hinzufügen der sechs Aktivitäten und deren Bezeichnung.	LK auf Werkzeugitem *ActionState,* LK in Zeichenebene und Benennen der Zustände.	Maus: 12 Tastatur: 12 Wechsel: 24
		48
04 - Hinzufügen der Entscheidungsknoten.	LK auf *Decision* und LK auf Zeichenebene.	Maus: 10 Tastatur: 0 Wechsel: 0
		10
05 - Einfügen des Sub-aktivität *Neues Konto anlegen*.	LK auf dafür vorgesehenes Icon, Auf-tragen in Zeichenbereich und Be-nennen.	Maus: 2 Tastatur: 2 Wechsel: 4
		8

Schritt	Vorgangsweise	Benötigte Aktionen
06 - Einfügen der Kontrollflusskanten.	LK auf die Kontrollflusskanten und LK auf Zeichenebene. Benennen der Kanten.	Maus: 34 Tastatur: 18 Wechsel: 36
		88
07 - Einfügen des Objektknotens *Bestellung*.	LK auf *SubactivityState* und LK auf Zeichenbereich und Benennen.	Maus: 2 Tastatur: 2 Wechsel: 4
		8
08 - Setzen des Aktivitätsendknotens.	LK auf Werkzeug *FinalState* und LK in Zeichenbereich.	Maus: 2 Tastatur: 0 Wechsel: 0
		2
GESAMT		**168**

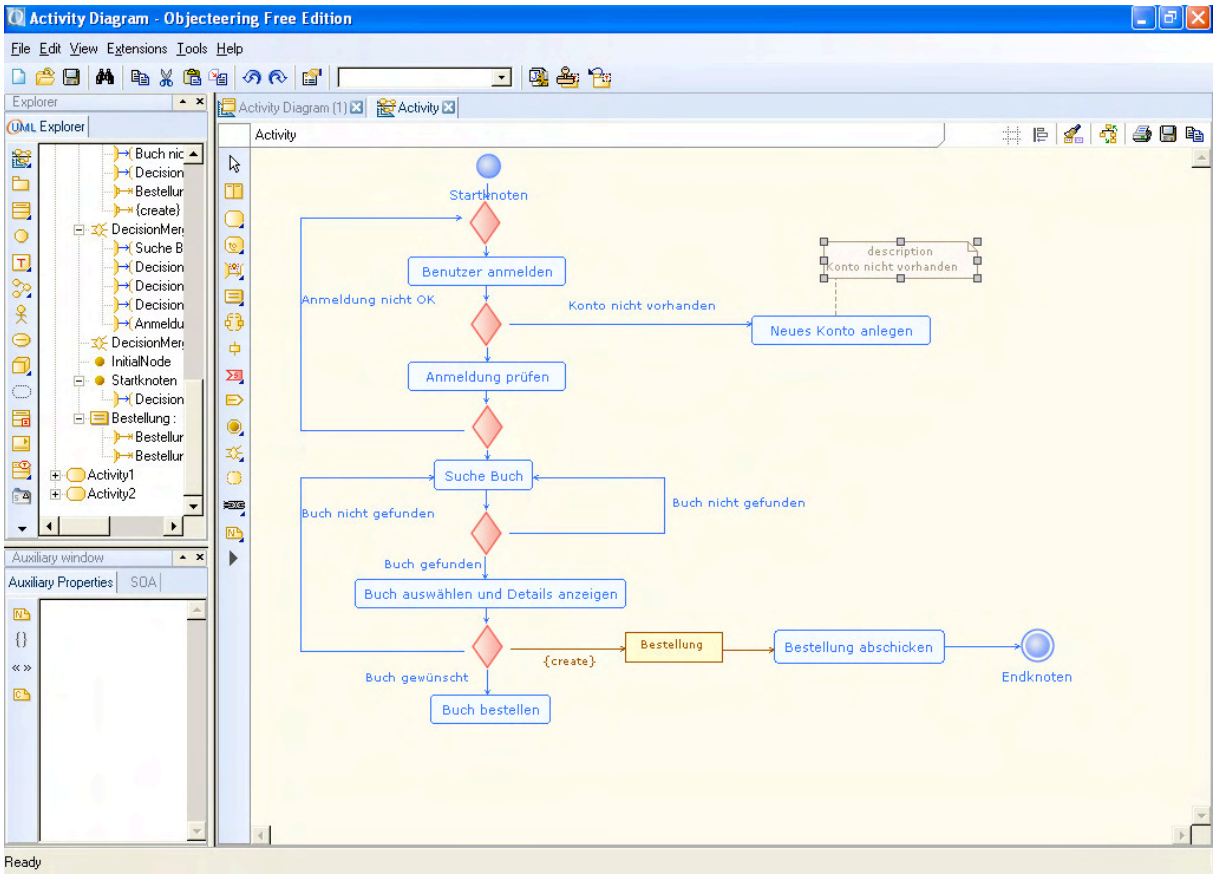
6.6.4. Aktivitätsdiagramm – Poseidon for UML



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Anlegen eines Aktivitätsdiagramms.	LK auf Icon *Neues Aktivitätsdiagramm*. Eingabe des Namens.	Maus: 1 Tastatur: 2 Wechsel: 4
		7
02 - Setzen des Startknotens.	LK aufs Startknotenwerkzeug und LK auf Zeichenebene. Eingabe der Bezeichnung.	Maus: 2 Tastatur: 2 Wechsel: 4
		8
03 - Hinzufügen der Aktivitäten und deren Benennung.	LK auf Icon *Aktion* und Eingabe der Bezeichnungen per Tastatur.	Maus: 14 Tastatur: 14 Wechsel: 28
		56
04 - Hinzufügen der Verzweigungsknoten und entfernen der automatischen Bezeichnungen.	LK auf Icon *Verzweigungsknoten* und entfernen der vom System automatisch eingefügten Bezeichnungen.	Maus: 10 Tastatur: 5 Wechsel: 10
		25
05 - Hinzufügen des Objektknotens *Bestellung*.	LK auf Icon *Objektknoten*. Hinzufügen der Bezeichnung.	Maus: 2 Tastatur: 2 Wechsel: 4
		8

Schritt	Vorgangsweise	Benötigte Aktionen
06 - Hinzufügen der Kanten.	Markieren der Aktivität durch „Berühren“ mit der Maus und LK auf Icon *Kante hinzufügen* und verbinden der jeweiligen Elemente. Benennen der Kanten durch LK und Tastatureingabe.	Maus: 24 Tastatur: 7 Wechsel: 14
		45
07 - Hinzufügen des Endknotens.	LK auf Endknotenwerkzeug, LK auf Zeichenbereich und Benennen des Knotens.	Maus: 2 Tastatur: 2 Wechsel: 4
		8
08 - Hinzufügen der Notiz.	LK auf *Notiz hinzufügen*.	Maus: 2 Tastatur: 2 Wechsel: 4
		8
GESAMT		**165**

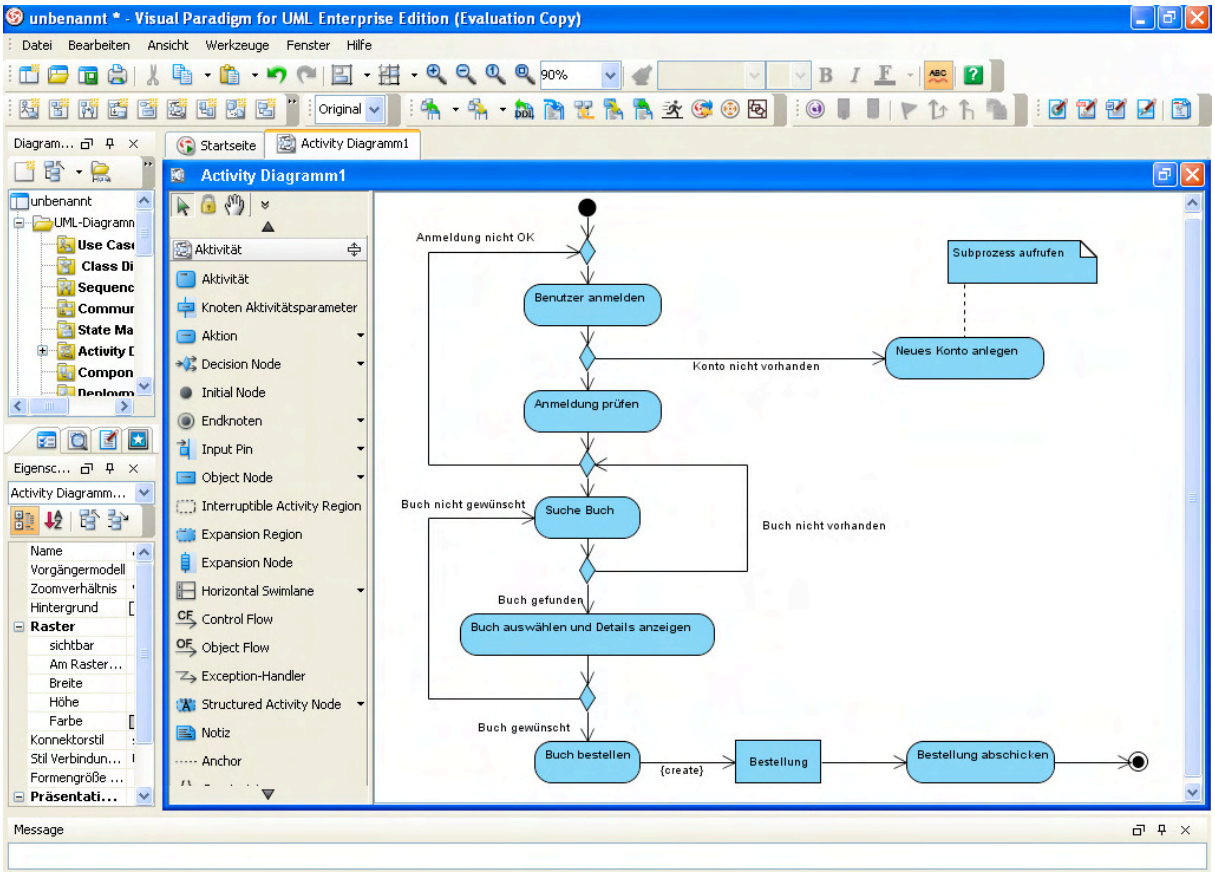
6.6.5. Aktivitätsdiagramm – Objecteering



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Neues Aktivitäts-diagramm anlegen.	LK im *UML Explorer* auf *Create an horizontal Activity Diagram* und anschließend LK auf das UML Project um diese zu benennen.	Maus: 3 Tastatur: 1 Wechsel: 2 6
02 - Einfügen des Initialknotens.	LK auf *Create an activity initial node,* LK in Zeichenebene und Beschriftung. Anschließend RK auf den Knoten *Representation options* wählen und LK auf *Show name*.	Maus: 4 Tastatur: 1 Wechsel: 2 7
03 - Hinzufügen der Aktivitäten und deren Benennung.	LK auf *Create an Aktion, LK in* Zeichenebene und Eingabe der Bezeichnungen per Tastatur.	Maus: 12 Tastatur: 6 Wechsel: 12 30
04 - Hinzufügen der Verzweigungsknoten.	LK auf Icon *Create a decision merge node* und LK auf die gewünschte Position.	Maus: 10 Tastatur: 0 Wechsel: 0 10

Schritt	Vorgangsweise	Benötigte Aktionen
05 - Hinzufügen des Objektknotens *Bestellung*.	LK auf Icon *Create a object node,* LK auf die gewünschte Position und hinzufügen der Bezeichnung.	Maus: 2 Tastatur: 1 Wechsel: 2 5
06 - Hinzufügen des Endknotens.	LK auf *Create an activity final node,* LK in Zeichenebene und Beschriftung. Anschließend RK auf den Knoten *Representation options* wählen und LK auf *Show name*.	Maus: 4 Tastatur: 1 Wechsel: 2 7
07 - Hinzufügen der Kanten.	LK auf *Create a control or object flow* und LK auf die zu verbindenden Aktionen/Objekte. Zur Benennung DK auf die Kante und Texteingabe. Anschließend RK auf den Knoten *Representation options* wählen und LK auf *Show name*.	Maus: 75 Tastatur: 7 Wechsel: 14 96
08 - Hinzufügen der Notiz.	LK auf *Create a note,* LK auf die Aktivität und LK auf die gewünschte Position. Zur Texteingabe DK auf die Notiz.	Maus: 5 Tastatur: 1 Wechsel: 2 8
GESAMT		**169**

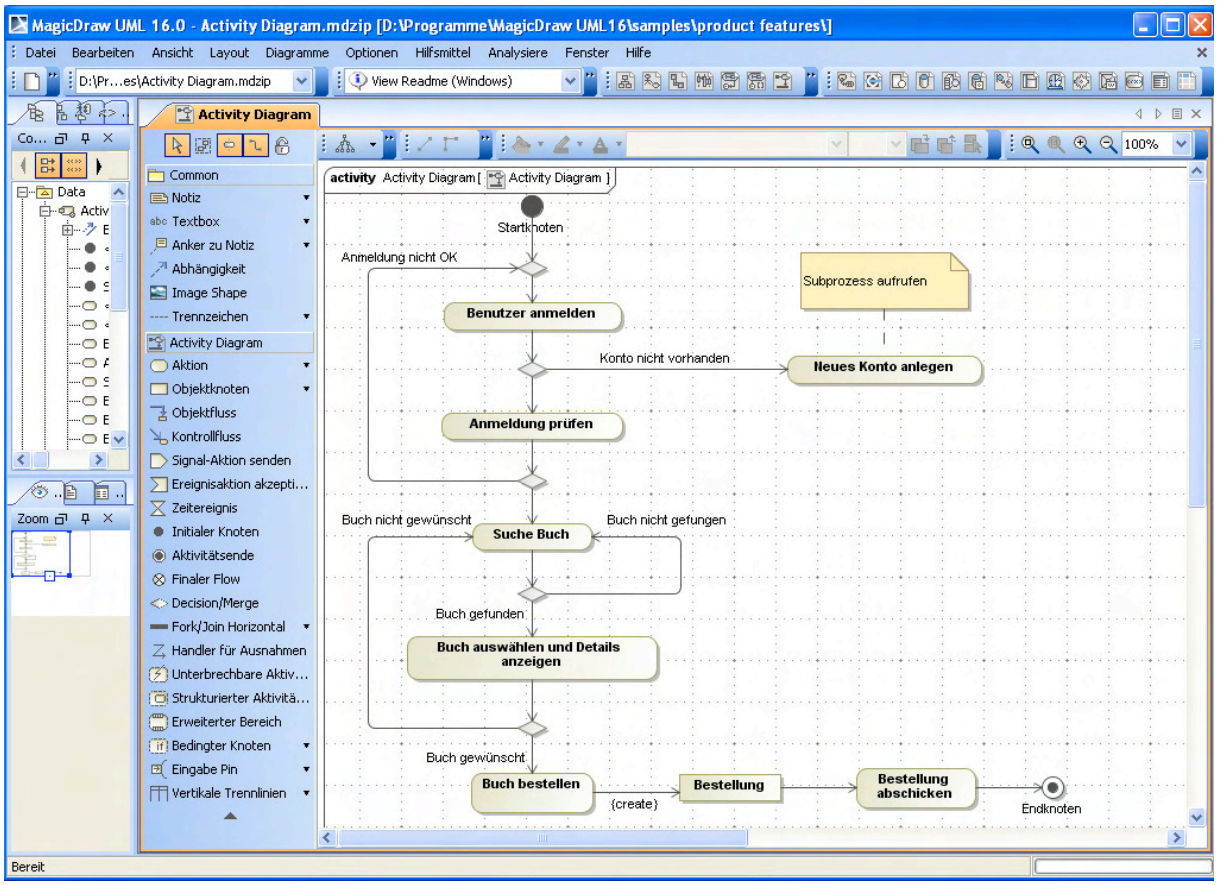
6.6.6. Aktivitätsdiagramm – Visual Paradigm



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Neues Aktivitäts-diagramm anlegen.	LK auf *Neues Activity Diagram*.	Maus: 1 Tastatur: 0 Wechsel: 0
		1
02 - Einfügen des Initi-alknotens.	Im Werkzeugbereich LK auf *Initial node* und ziehen in den Zeichenbereich.	Maus: 2 Tastatur: 0 Wechsel: 0
		2
03 - Hinzufügen der Aktivitäten und deren Benennung.	LK auf *Aktivität,* ziehen in die Zeichenebene und Eingabe der Bezeichnungen per Tastatur.	Maus: 14 Tastatur: 7 Wechsel: 14
		35
04 - Hinzufügen der Verzweigungsknoten.	Im Werkzeugbereich LK auf *Decision node* und ziehen in den Zeichenbereich.	Maus: 10 Tastatur: 0 Wechsel: 0
		10
05 - Hinzufügen des Objektknotens *Bestel-lung*.	LK auf Icon *Create a object node,* LK auf die gewünschte Position und hin-zufügen der Bezeichnung.	Maus: 2 Tastatur: 1 Wechsel: 2
		5

Schritt	Vorgangsweise	Benötigte Aktionen
06 - Hinzufügen des Endknotens.	Im Werkzeugbereich LK auf *Endknoten* und ziehen in den Zeichenbereich.	Maus: 2 Tastatur: 0 Wechsel: 0
		2
07 - Hinzufügen der Kanten.	LK auf *control flow* bzw. *object flow*, LK auf die Aktion bzw. das Objekt und ziehen zum Ziel. Anschließend Eingabe der Bezeichnung..	Maus: 63 Tastatur: 7 Wechsel: 14
		84
08 - Hinzufügen der Notiz.	LK auf *Notiz,* ziehen auf die gewünschte Position und Eingabe des Textes. LK auf *Anchor,* LK auf die Aktivität und ziehen zur Notiz	Maus: 5 Tastatur: 1 Wechsel: 2
		8
GESAMT		**147**

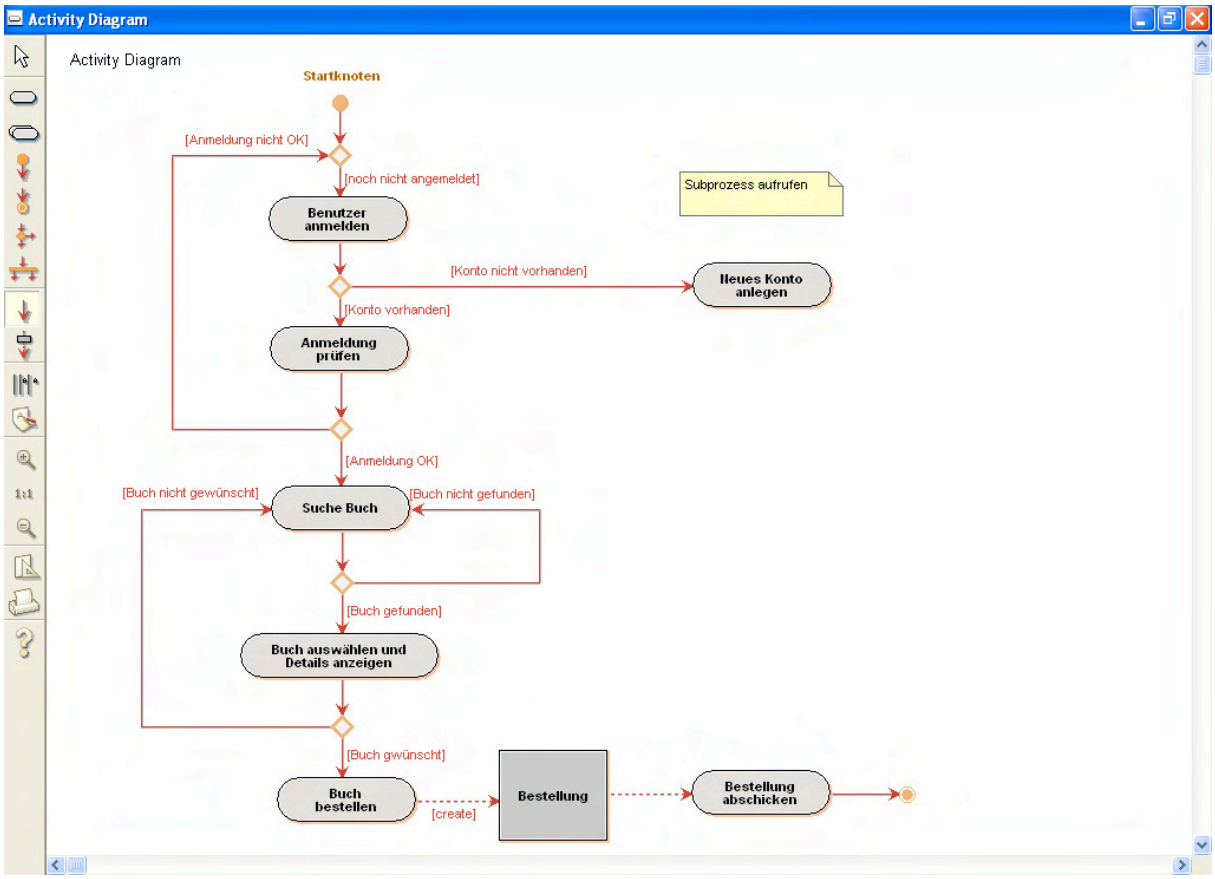
6.6.7. Aktivitätsdiagramm – Magic Draw



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Neues Aktivitäts-diagramm anlegen.	LK auf Icon *Aktivitätsdiagramme*. Diagramm benennen und bestätigen.	Maus: 1 Tastatur: 2 Wechsel: 2
		5
02 - Einfügen des Initi-alknotens.	Im Werkzeugbereich LK auf *Initialer Knoten,* LK auf die gewünschte Position und Texteingabe.	Maus: 2 Tastatur: 1 Wechsel: 2
		5
03 - Hinzufügen der Aktivitäten und deren Benennung.	LK auf *Aktion,* LK in die Zeichenebene und Eingabe der Bezeichnungen per Tastatur.	Maus: 14 Tastatur: 7 Wechsel: 14
		35
04 - Hinzufügen der Verzweigungsknoten.	Im Werkzeugbereich LK auf *Decisio/Merge* und LK auf die gewünschte Position.	Maus: 10 Tastatur: 0 Wechsel: 0
		10

Schritt	Vorgangsweise	Benötigte Aktionen
05 - Hinzufügen des Objektknotens *Bestellung*.	LK auf Icon *Objektknoten,* LK auf die gewünschte Position und hinzufügen der Bezeichnung.	Maus: 2 Tastatur: 1 Wechsel: 2 5
06 - Hinzufügen des Endknotens.	Im Werkzeugbereich LK auf *Aktivitätsende,* LK auf die gewünschte Position und Texteingabe.	Maus: 2 Tastatur: 1 Wechsel: 2 5
07 - Hinzufügen der Kanten.	LK auf *Kontrollfluss* bzw. *Objektfluss*, LK auf die zu verbindenden Aktionen bzw. Objekte. Anschließend Eingabe der Bezeichnung.	Maus: 57 Tastatur: 7 Wechsel: 14 78
08 - Hinzufügen der Notiz.	LK auf die *Aktivität*, LK auf *Anker zur Notiz,* LK auf die gewünschte Position und Texteingabe.	Maus: 4 Tastatur: 1 Wechsel: 2 7
GESAMT		**150**

6.6.8. Aktivitätsdiagramm – ObjectIF

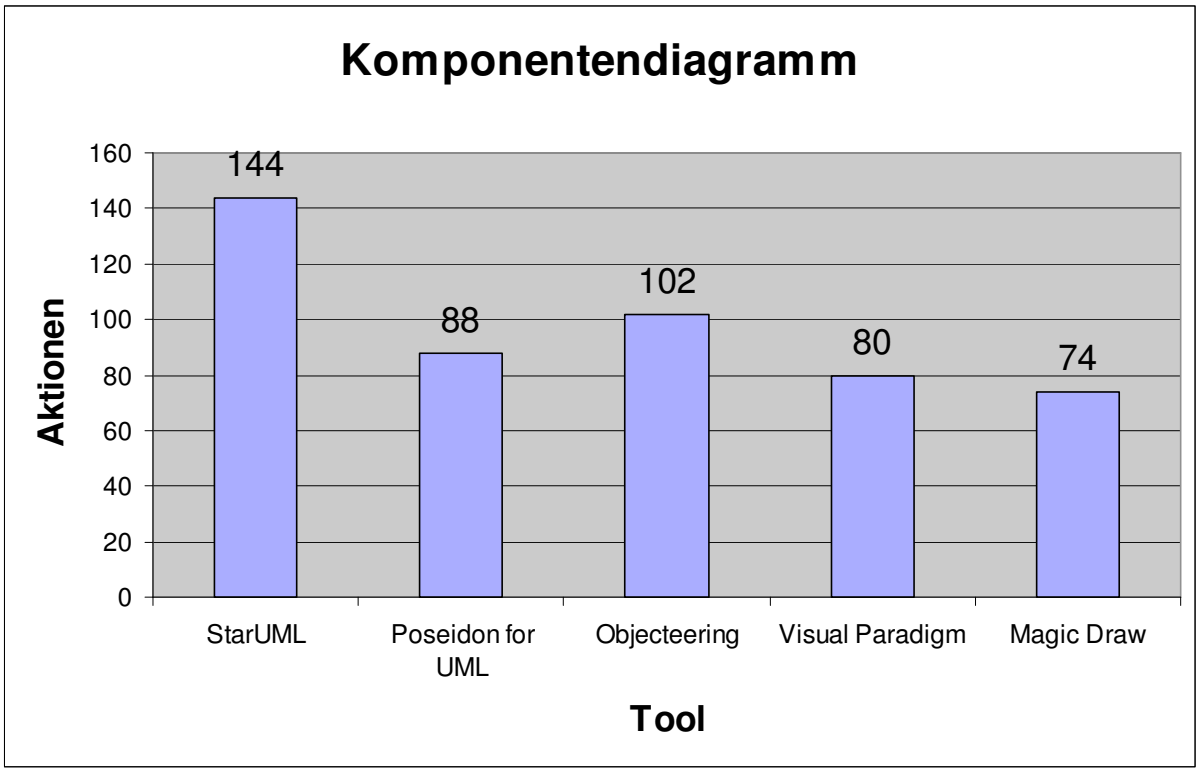


Schritt	Vorgangsweise	Benötigte Aktionen
01 – Neues Aktivitäts-diagramm anlegen.	LK auf Icon Design. RK in den leeren Bereich. Maus über *Anlegen* und LK auf *Aktivitätsdiagramm*. Name vergeben und bestätigen.	Maus: 3 Tastatur: 2 Wechsel: 2 7
02 - Einfügen des Initialknotens.	Im Werkzeugbereich LK auf *Startzustand anlegen* und LK auf die gewünschte Position. RK auf den Knoten, LK auf Eigenschaften und Texteingabe.	Maus: 4 Tastatur: 1 Wechsel: 2 7
03 - Hinzufügen der Aktivitäten und deren Benennung.	Im Werkzeugbereich LK auf *Aktion anlegen,* LK in die Zeichenebene und Eingabe der Bezeichnungen per Tastatur.	Maus: 8 Tastatur: 7 Wechsel: 14 29
04 - Hinzufügen der Verzweigungsknoten.	Im Werkzeugbereich LK auf *Entscheidungsknoten anlegen* und LK auf die gewünschte Position.	Maus: 6 Tastatur: 0 Wechsel: 0 6

Schritt	Vorgangsweise	Benötigte Aktionen
05 - Hinzufügen des Objektknotens *Bestellung*.	LK auf Icon *Objektfluss anlegen,* LK auf die Aktivitäten und Eingabe der Bezeichnung.	Maus: 3 Tastatur: 1 Wechsel: 2 6
06 - Hinzufügen des Endknotens.	Im Werkzeugbereich LK auf *Endzustand anlegen* und LK auf die gewünschte Position.	Maus: 2 Tastatur: 0 Wechsel: 0 2
07 - Hinzufügen der Kanten.	LK auf *Transaktion anlegen*, LK auf die zu verbindenden Aktivitäten. Anschließend Eingabe der Bezeichnung.	Maus: 33 Tastatur: 10 Wechsel: 20 63
08 - Hinzufügen der Notiz.	LK auf Notiz anlegen, LK auf die gewünschte Position, DK und Texteingabe.	Maus: 4 Tastatur: 1 Wechsel: 2 7
GESAMT		**127**

6.7 Erstellung des Komponentendiagramms

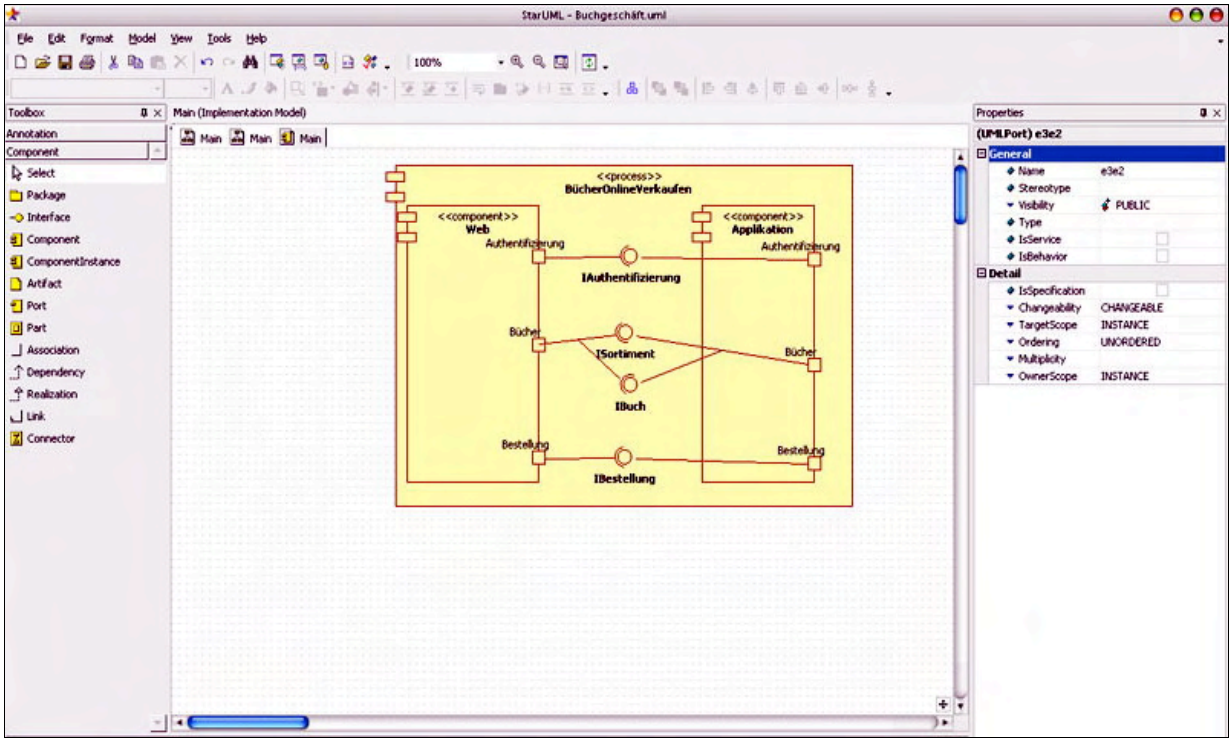
6.7.1. Zusammenfassung der Ergebnisse:



6.7.2. Interpretation der Ergebnisse

Die größten Unterschiede ergeben sich beim hinzufügen von Interfaces. Bei StarUML können die einzelnen Ports nur auf der linken Seite einer Komponente hinzugefügt werden, was die Lesbarkeit bei größeren Diagrammen erschweren kann. Bei Magic Draw muss man vor allem beim Hinzufügen von Ports aufpassen. Man muss ihnen einen Klassentyp zuweisen, damit in weiterer Folge eine Verbindung zwischen dem Port und einem Interface überhaupt möglich ist. Mit ObjectIF ist das Design eines Komponentendiagramms nicht möglich.

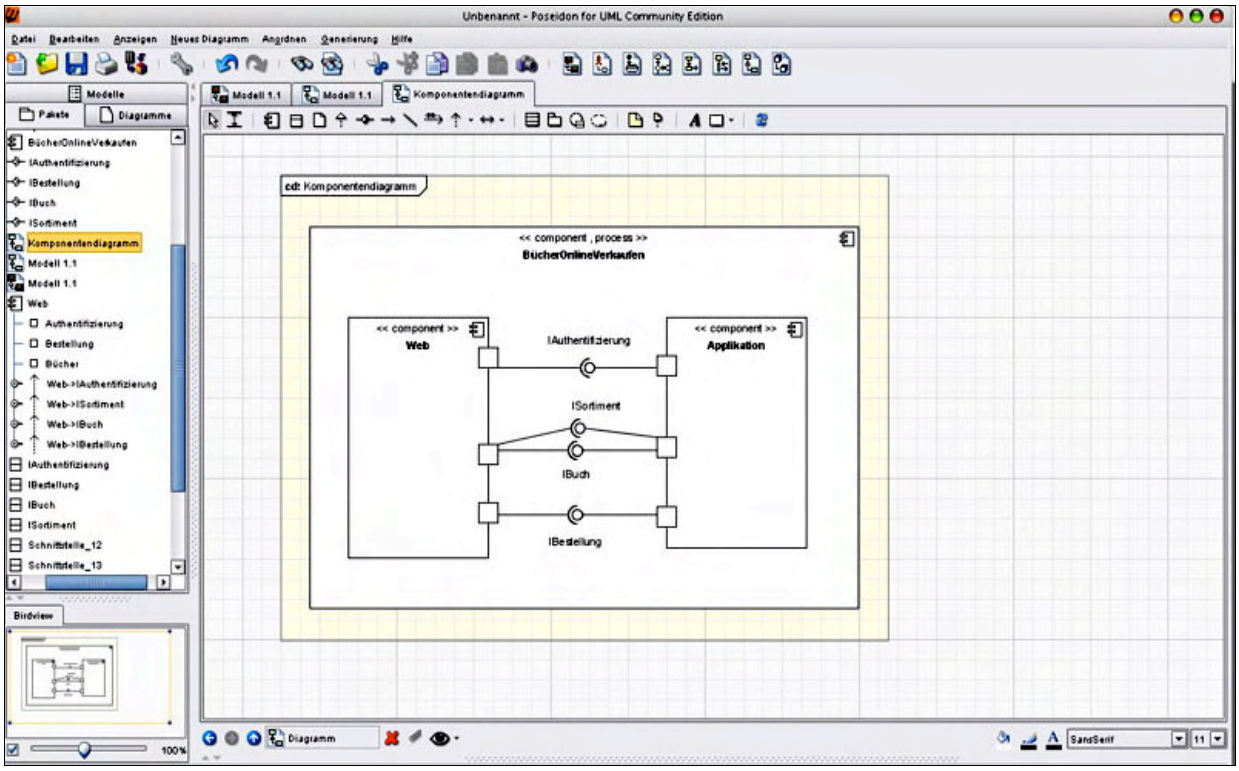
6.7.3. Komponentendiagramm – StarUML



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Anlegen des neuen Komponentendiagramms.	Aufrufen des Modellexplorers und wechseln in der Baumstruktur zum Komponentendiagramm.	Maus: 3 Tastatur: 0 Wechsel: 0
		3
02 - Hinzufügen der drei Komponenten und setzen der Stereotypen.	LK auf Komponentenwerkzeug und ziehen der Komponenten. Mit LK in die Eigenschaften wechseln und durch LK setzen der Stereotypen.	Maus: 21 Tastatur: 8 Wechsel: 16
		45
03 - Hinzufügen der sechs Ports und die Benennung der Ports.	LK auf Port-Werkzeug. Hinzufügen des Ports zur Komponente. Eingabe der Bezeichnung und bestätigen durch Tastatureingabe.	Maus: 12 Tastatur: 12 Wechsel: 24
		48
04 - Anlegen der vier Interfaces.	Interface aus Werkzeugleiste auswählen, hinzufügen zum Diagramm durch LK und benennen.	Maus: 8 Tastatur: 8 Wechsel: 16
		32

Schritt	Vorgangsweise	Benötigte Aktionen
05 - Hinzufügen der Verbindungen zwischen den einzelnen Ports und der Interfaces.	LK auf *Realization*-Werkzeug um ein angebotenes Interface hinzuzufügen. Ziehen der Verbindung vom gewünschten Port aus zum Interface. Ports der Komponente Web mit Hilfe des *Dependency*-Werkzeugs mit dem angebotenen Interface der Komponente Applikation verbinden.	Maus: 16 Tastatur: 0 Wechsel: 0
		16
GESAMT		**144**

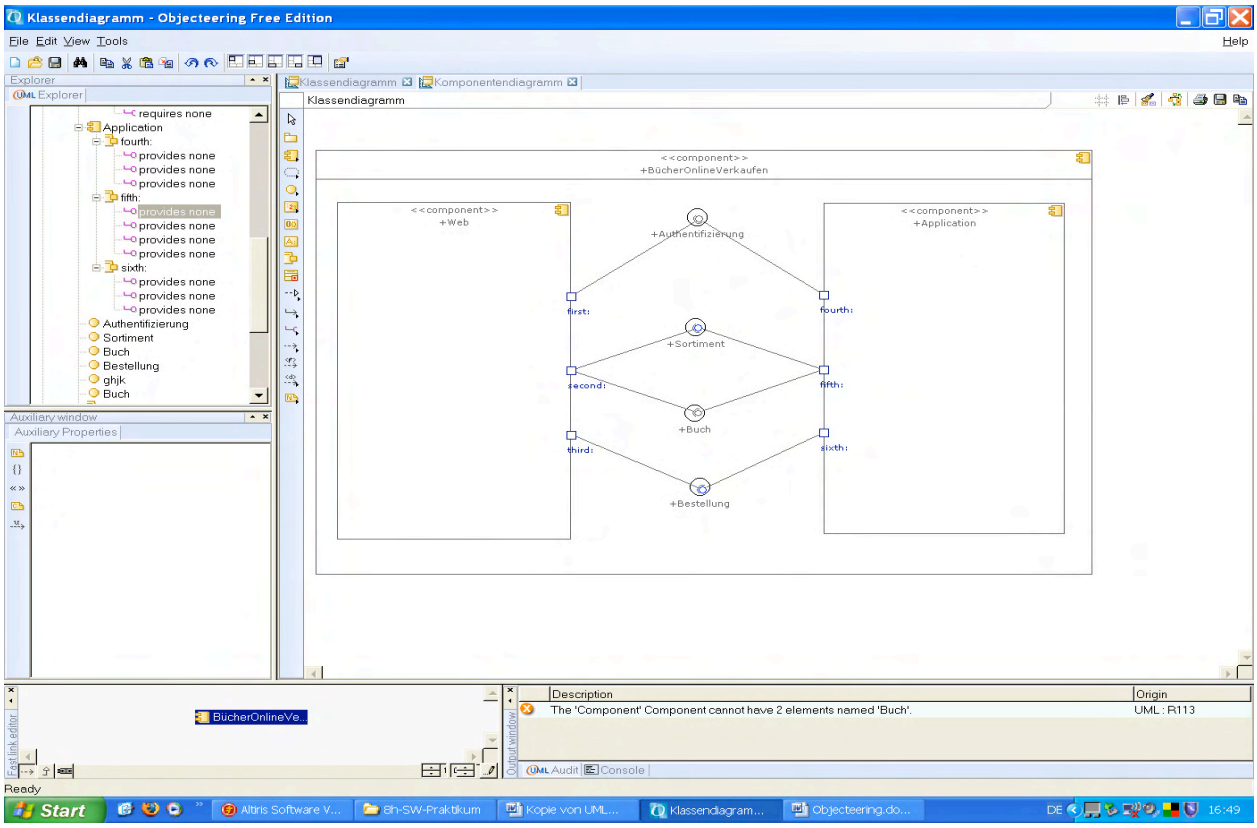
6.7.4. Komponentendiagramm – Poseidon for UML



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Anlegen des neuen Komponenten-diagramms.	LK auf Icon *Neues Komponenten-diagramm* und Eingabe der Be-zeichnung.	Maus: 1 Tastatur: 2 Wechsel: 4 7
02 - Hinzufügen der drei Komponenten und set-zen der Stereotypen.	LK auf Komponentenicon und LK auf Zeichenebene. Eingabe der Bezeichnung. Wechseln in Eigenschaften, RK auf Stereotypen und LK auf *set stereotypes*. Auswahl des Stereotyps mit LK hinzufügen.	Maus: 11 Tastatur: 6 Wechsel: 12 29
03 - Hinzufügen der sechs Ports und die Benennung der Ports.	LK auf *Port-Hinzufügen*. Eingabe der Bezeichnungen unter den Portseigenschaften.	Maus: 8 Tastatur: 8 Wechsel: 16 32
04 - Hinzufügen angebo-tenen Schnittstellen.	LK um gewünschten Port auszuwählen und LK auf dafür vorgesehenen Icon.	Maus: 8 Tastatur: 0 Wechsel: 0 8

Schritt	Vorgangsweise	Benötigte Aktionen
05 - Verbinden der Ports mit den Interfaces.	LK für Portauswahl, LK auf Icon und verbinden der Schnittstellen durch drag-and-drop.	Maus: 12 Tastatur: 0 Wechsel: 0
		12
GESAMT		**88**

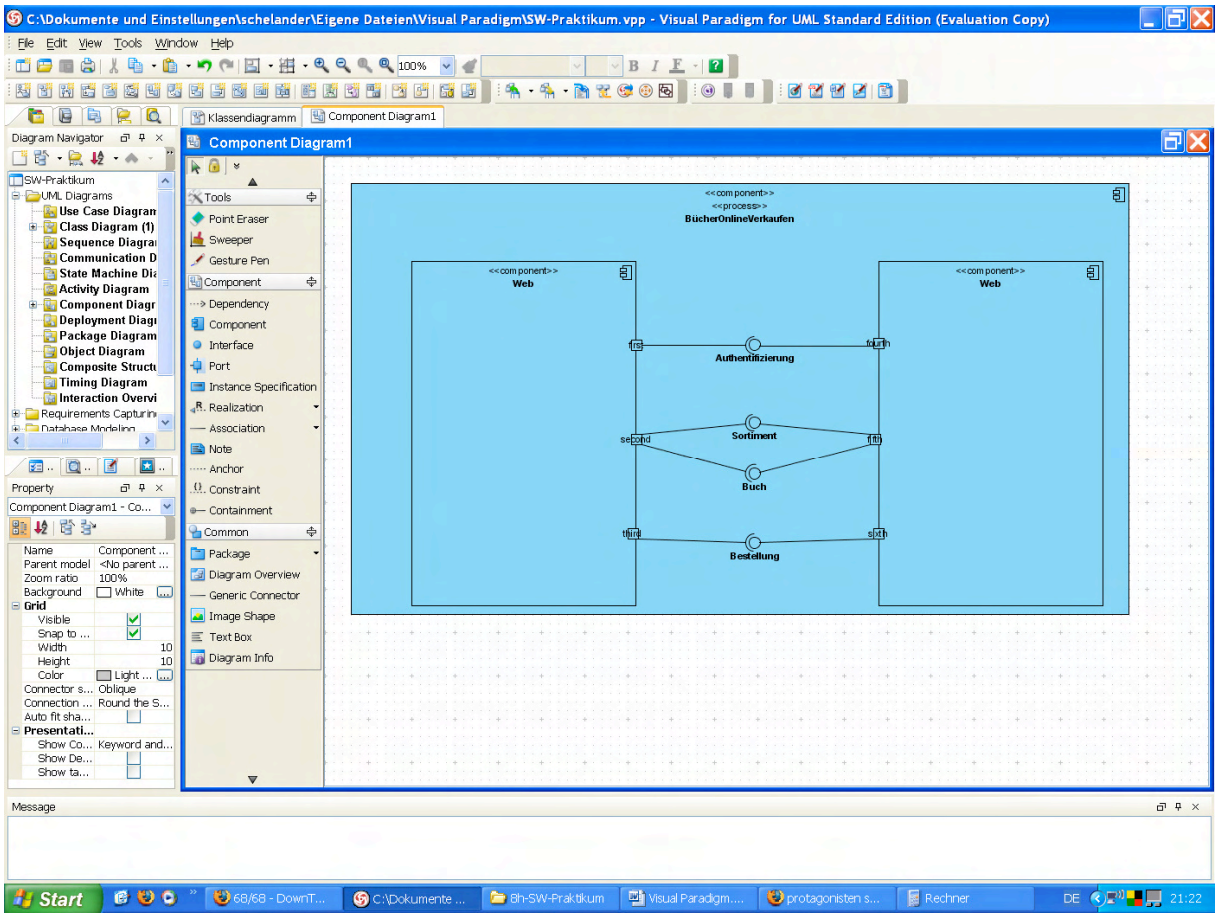
6.7.5. Komponentendiagramm – Objecteering



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Anlegen des neuen Komponentendiagramms.	LK auf Icon *Create a UML 2.0 static class diagram*.	Maus: 3 Tastatur: 0 Wechsel: 0
		3
02 - Hinzufügen der drei Komponenten und setzen der Stereotypen.	LK auf Icon *Create a component* und LK im Zeichenbereich. Mit LK in die Eigenschaften wechseln und Bezeichnung eingeben.	Maus: 8 Tastatur: 4 Wechsel: 3
		15
03 - Hinzufügen der sechs Ports und die Benennung der Ports.	LK auf *Create a port*. LK im Zeichenbereich. Eingabe der Bezeichnung.	Maus: 12 Tastatur: 6 Wechsel: 11
		29
04 - Anlegen der vier Interfaces.	LK auf Icon *Create a new interface*. Benennung. Bestätigung mit Enter.	Maus: 8 Tastatur: 8 Wechsel: 15
		31

Schritt	Vorgangsweise	Benötigte Aktionen
05 - Hinzufügen der Verbindungen zwischen den einzelnen Ports und der Interfaces.	LK auf *Create a required interface* bzw. *Create a provided interface* und Festlegen der Endpunkte im Zeichenbereich.	Maus: 24 Tastatur: 0 Wechsel: 0
		24
GESAMT		**102**

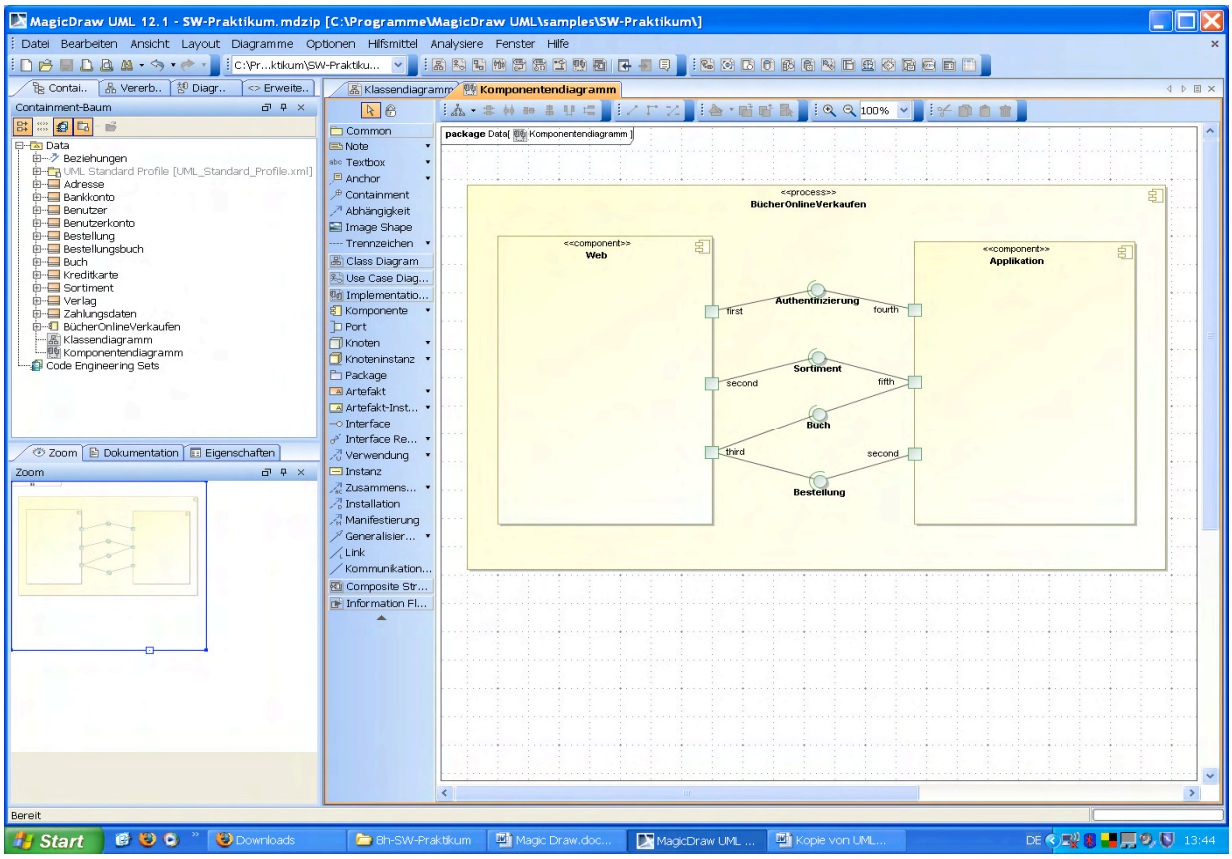
6.7.6. Komponentendiagramm – Visual Paradigm



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Anlegen des neuen Komponentendiagramms.	LK auf Icon *New Component Diagram*.	Maus: 1 Tastatur: 0 Wechsel: 0 1
02 - Hinzufügen der drei Komponenten und setzen der Stereotypen.	LK auf Icon *Component*, in den Zeichenbereich ziehen, beschriften, Stereotypen über RK und Kontektmenü hinzufügen.	Maus: 10 Tastatur: 3 Wechsel: 6 19
03 - Hinzufügen der sechs Ports und die Benennung der Ports.	LK auf *Port.* LK auf die Komponente, der der Port hinzugefügt werden soll.	Maus: 12 Tastatur: 6 Wechsel: 11 29
04 - Anlegen der vier Interfaces.	LK auf Icon *Interface*. Benennung. Bestätigung mit Enter.	Maus: 8 Tastatur: 8 Wechsel: 7 23

Schritt	Vorgangsweise	Benötigte Aktionen
05 - Hinzufügen der Verbindungen zwischen den einzelnen Ports und der Interfaces.	LK auf *Association* bzw. *Realization*. Verbindung durch ziehen von einem gewünschen Verbindungspunkt zum anderen.	Maus: 8 Tastatur: 0 Wechsel: 0 8
GESAMT		**80**

6.7.7. Komponentendiagramm – Magic Draw

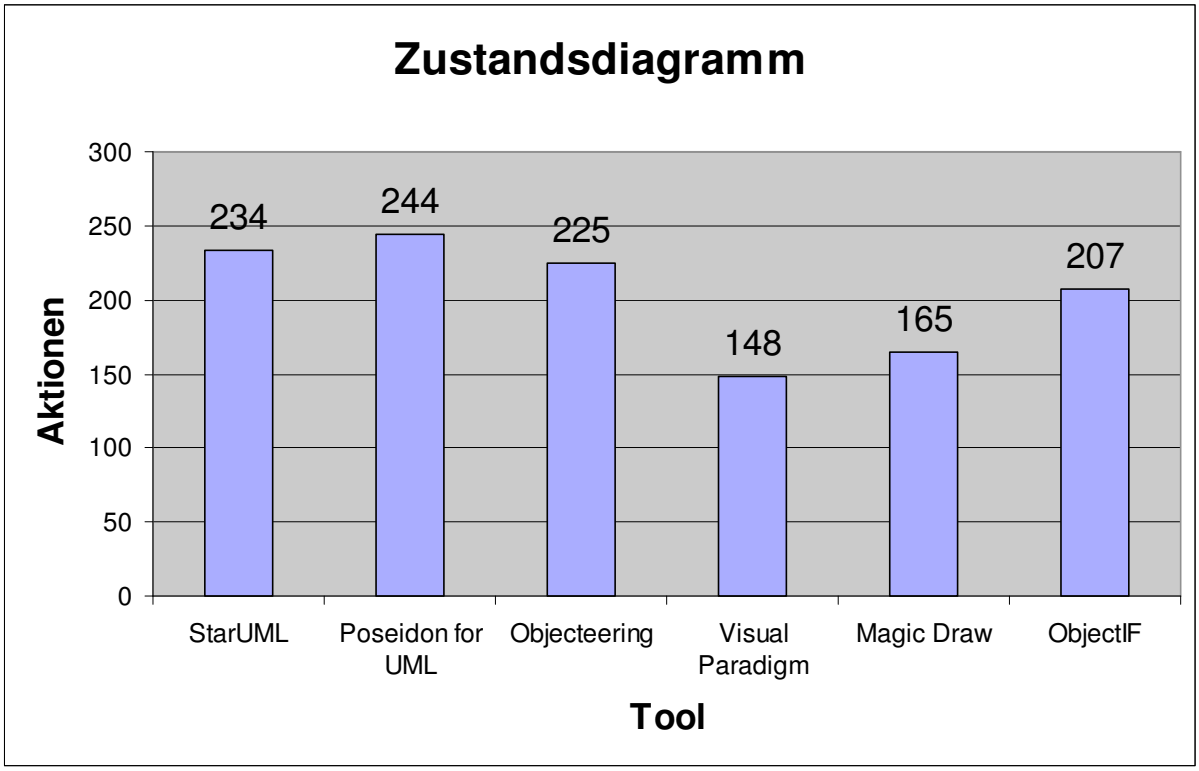


Schritt	Vorgangsweise	Benötigte Aktionen
01 - Anlegen des neuen Komponentendiagramms.	LK auf Icon *Implementationsdiagramm*. Name vergeben. Bestätigung mit Enter.	Maus: 1 Tastatur: 2 Wechsel: 1 4
02 - Hinzufügen der drei Komponenten und setzen der Stereotypen.	LK auf *Komponente*, im Zeichenbereich ziehen, beschriften, Stereotypen über RK und Kontektmenü hinzufügen.	Maus: 9 Tastatur: 3 Wechsel: 5 17
03 - Hinzufügen der sechs Ports und die Benennung der Ports.	LK auf *Port.* LK auf die gewünschte Komponente. Klasse als Typ wählen, damit später ein angebotenes Interface vom Port weggezeichnet werden kann. Port benennen.	Maus: 9 Tastatur: 6 Wechsel: 11 26
04 - Anlegen der vier Interfaces.	LK auf Icon *Interface*. Benennung.	Maus: 8 Tastatur: 4 Wechsel: 7 19

Schritt	Vorgangsweise	Benötigte Aktionen
05 - Hinzufügen der Verbindungen zwischen den einzelnen Ports und der Interfaces.	LK auf *Verwendung* bzw. *Interface Realization*. Verbindung durch ziehen von einem gewünschten Verbindungspunkt zum anderen.	Maus: 8 Tastatur: 0 Wechsel: 0 8
GESAMT		**74**

6.8 Erstellung des Zustandsdiagramms

6.8.1. Zusammenfassung der Ergebnisse



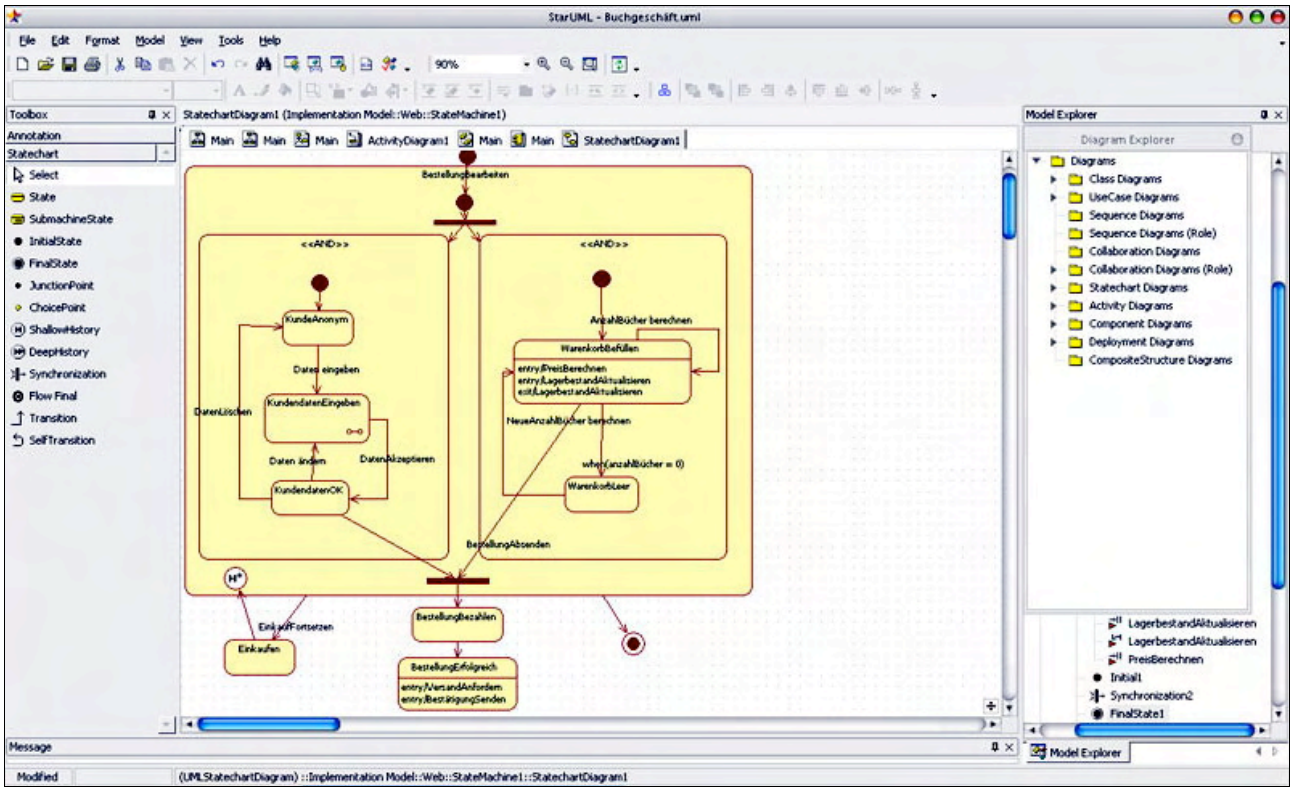
6.8.2. Interpretation der Ergebnisse

Bezüglich der Anzahl der Interaktionsschritte liegen beim Zustandsdiagramm StarUML und Poseidon for UML fast gleich auf. Mit beiden Werkzeugen lassen sich Zustandsdiagramme problemlos erstellen.

Am schnellsten funktioniert die Erstellung eines Zustandsdiagramms mit Visual Paradigm.

Das Design von zusammengesetzten Zuständen ist mit ObjectIF nicht so möglich wie bei den anderen Tools, weil ObjectIF innerhalb eines Zustands keine weiteren Notationselemente mehr erlaubt. Man kann nur Subzustandsdiagramme anlegen. Es können nur Elemente des gleichen Subzustands miteinander verbunden werden. So war zum Beispiel die bei allen anderen Tools eingefügte Transaktion vom Zustand KundendatenOK zur Vereinigung (die im Superzustand liegt) nicht möglich. Wenn man alles sehen will, muss man zwischen den Subdiagrammen, die alle zum gleichen Hauptdiagramm gehören, hin- und herspringen. Für Zustandsdiagramme ist ObjectIF demnach keinesfalls empfehlenswert.

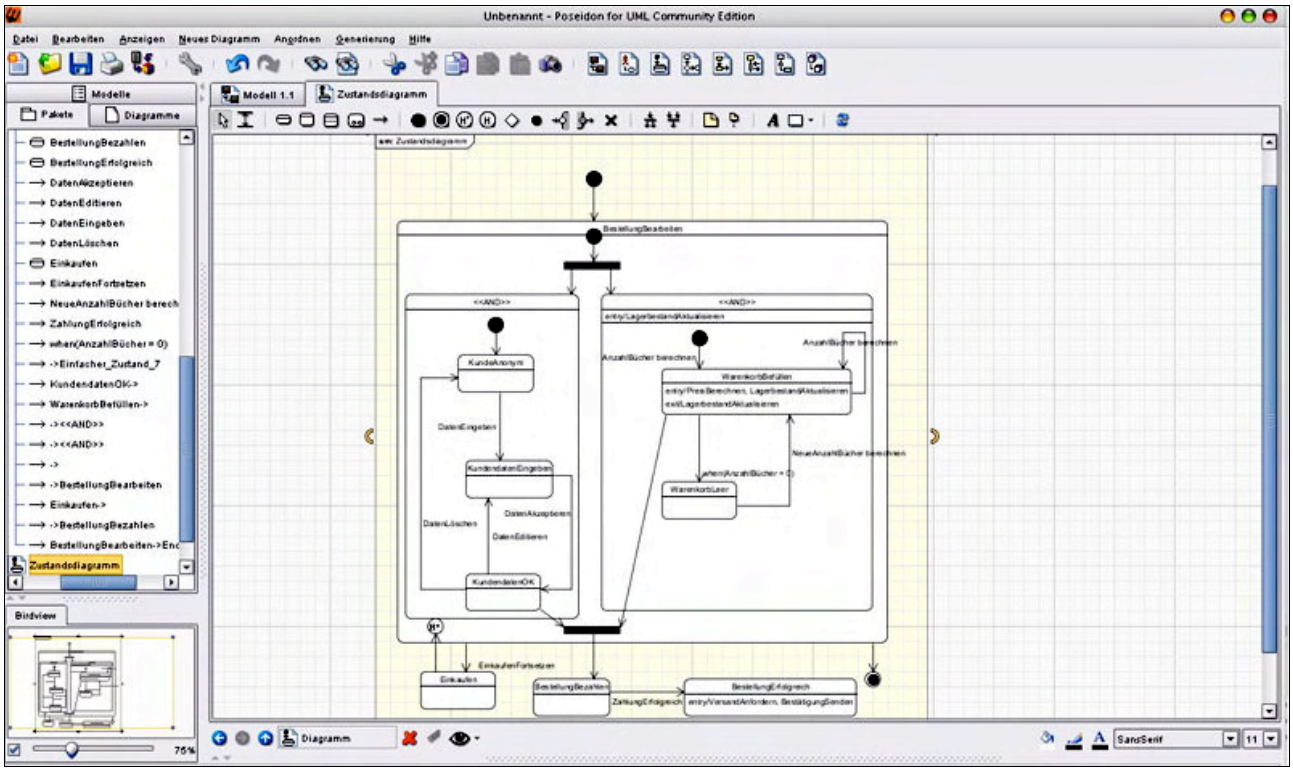
6.8.3. Zustandsdiagramm – StarUML



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Anlegen eines Zustandsdiagramms.	LK im Menü auf *Model* und dann auf *Add Diagram*. LK auf *StatechartDiagram.*	Maus: 3 Tastatur: 0 Wechsel: 0
		3
02 - Hinzufügen der drei großen Zustände.	LK auf Werkzeug *State* und LK auf Zeichenebene. Eingabe der Bezeichnung. Hinzufügen der Stereotypen.	Maus: 10 Tastatur: 6 Wechsel: 12
		28
03 - Hinzufügen des Zustandes *WarenkorbBefüllen*.	LK auf *State*, LK in Zeichenebene. Hinzufügen der Bezeichnung und Anlegen der Eingangs- und Ausgangsaktionen.	Maus: 5 Tastatur: 4 Wechsel: 8
		17
04 - Hinzufügen des Zustandes *WarenkorbLeer*.	LK auf *State*, LK in Zeichenbereich und bezeichnen des Zustandes.	Maus: 2 Tastatur: 2 Wechsel: 4
		8
05 - Setzen des Anfangszustandes.	LK auf *InitialState* und LK auf Zeichenbereich.	Maus: 2 Tastatur: 0 Wechsel: 0
		2

Schritt	Vorgangsweise	Benötigte Aktionen
06 - Hinzufügen der Verbindungen.	LK auf Pfeilwerkzeug und ziehen der Verbindungen. Bezeichnungen hinzufügen.	Maus: 14 Tastatur: 8 Wechsel: 16 38
07 - Hinzufügen der Zustände *KundeAnonym, KundendatenEingeben, KundendatenOK*.	LK auf *State*, LK in Zeichenbereich und bezeichnen des Zustandes.	Maus: 6 Tastatur: 6 Wechsel: 12 24
08 - Setzen des Anfangszustandes.	LK auf *InitialState* und LK auf Zeichenbereich.	Maus: 2 Tastatur: 0 Wechsel: 0 2
09 - Hinzufügen der Transitionen.	LK auf Pfeilwerkzeug und ziehen der Verbindungen. Bezeichnungen hinzufügen.	Maus: 14 Tastatur: 8 Wechsel: 16 38
10 - Auftragen der beiden Synchronisationen.	LK auf *Synchronization* und LK auf Zeichenbereich.	Maus: 4 Tastatur: 0 Wechsel: 0 4
11 - Einfügen der Zustände *BestellungBezahlen, BestellungErfolgreich* und *Einkaufen*.	LK auf *State*, LK in Zeichenbereich und bezeichnen des Zustandes. Anlegen der Entrys.	Maus: 8 Tastatur: 6 Wechsel: 12 26
12 - Setzen des restlichen Anfangszustände und des Endzustandes.	LK auf Werkzeugitem und LK in Zeichenbereich.	Maus: 8 Tastatur: 0 Wechsel: 0 8
13 - Hinzufügen der restlichen Verbindungen.	LK auf Pfeilwerkzeug und ziehen der Verbindungen. Bezeichnungen hinzufügen.	Maus: 24 Tastatur: 4 Wechsel: 8 36
GESAMT		**234**

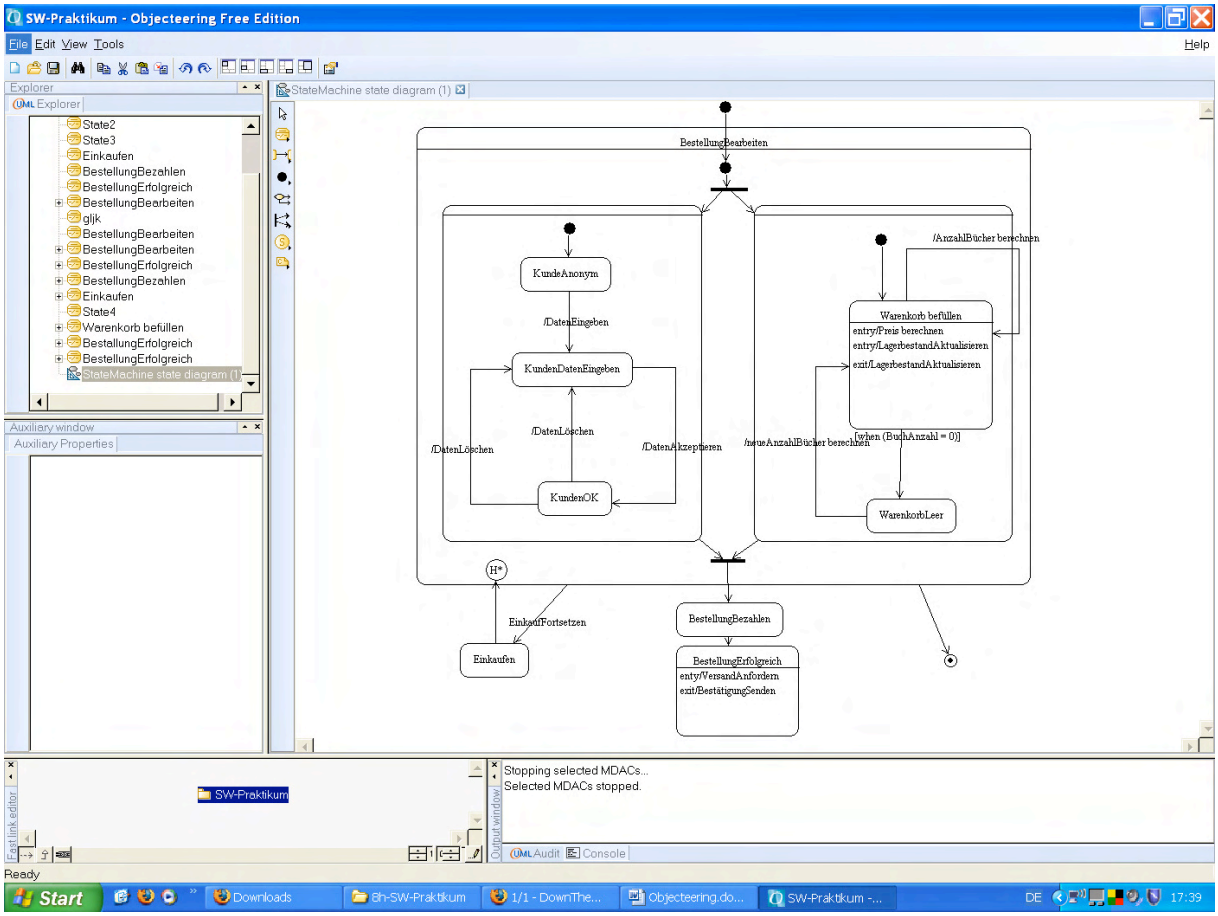
6.8.4. Zustandsdiagramm – Poseidon for UML



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Anlegen eines neuen Zustandsdiagramms.	LK auf Icon *Neues Zustandsdiagramm*. Eingabe der Bezeichnung.	Maus: 2 Tastatur: 2 Wechsel: 4
		8
02 - Hinzufügen der drei großen Zustände.	LK auf Werkzeug *Kompositionszustand*, LK in Zeichenbereich. Eingabe der Bezeichnung.	Maus: 6 Tastatur: 6 Wechsel: 12
		24
03 - Hinzufügen des Zustandes *WarenkorbBefüllen*.	LK auf *Zustand* LK in Zeichenbereich, Vergeben der Bezeichnung. Angeben der Eingangs- und Ausgangsaktionen unter Eigenschaften.	Maus: 12 Tastatur: 4 Wechsel: 8
		24
04 - Hinzufügen des Zustandes *Warenkorb-Leer*.	LK auf Icon *Zustand* und LK ins Diagramm. Vergabe der Bezeichnung.	Maus: 4 Tastatur: 4 Wechsel: 8
		16
05 - Setzen des Anfangszustandes.	LK auf *Startzustand* und LK auf Zeichenbereich.	Maus: 2 Tastatur: 0 Wechsel: 0
		2

Schritt	Vorgangsweise	Benötigte Aktionen
06 - Hinzufügen der Transitionen.	Berühren des Zustandes, aus dem die Transition ausgeht, LK auf Pfeilicon und ziehen der Transition.	Maus: 6 Tastatur: 8 Wechsel: 16 30
07 - Hinzufügen der Zustände *KundeAnonym, KundendatenEingeben, KundendatenOK*.	LK auf Icon *Zustand* und LK ins Diagramm. Vergabe der Bezeichnung.	Maus: 12 Tastatur: 12 Wechsel: 24 48
08 - Setzen des Anfangszustandes.	LK auf *Startzustand* und LK auf Zeichenbereich.	Maus: 2 Tastatur: 0 Wechsel: 0 2
09 - Hinzufügen der Transitionen.	Berühren des Zustandes, aus dem die Transition ausgeht, LK auf Pfeilicon und ziehen der Transition. Benennen der Verbindungen.	Maus: 7 Tastatur: 8 Wechsel: 16 31
10 - Auftragen der beiden Synchronisationen	LK auf *Gabelungszustand* und LK in Zeichenebene.	Maus: 4 Tastatur: 0 Wechsel: 0 4
11 - Einfügen der Zustände *BestellungBezahlen, BestellungErfolgreich* und *Einkaufen*.	LK auf Icon *Zustand* und LK ins Diagramm. Vergabe der Bezeichnung.	Maus: 8 Tastatur: 5 Wechsel: 10 23
12 - Setzen des restlichen Anfangszustände und des Endzustandes.	LK auf Werkzeugitem und LK in Zeichenbereich.	Maus: 8 Tastatur: 0 Wechsel: 0 8
13 - Hinzufügen der restlichen Transitionen..	Berühren des Zustandes, aus dem die Transition ausgeht, LK auf Pfeilicon und ziehen der Transition. Benennen der Verbindungen.	Maus:12 Tastatur: 4 Wechsel: 8 24
GESAMT		**244**

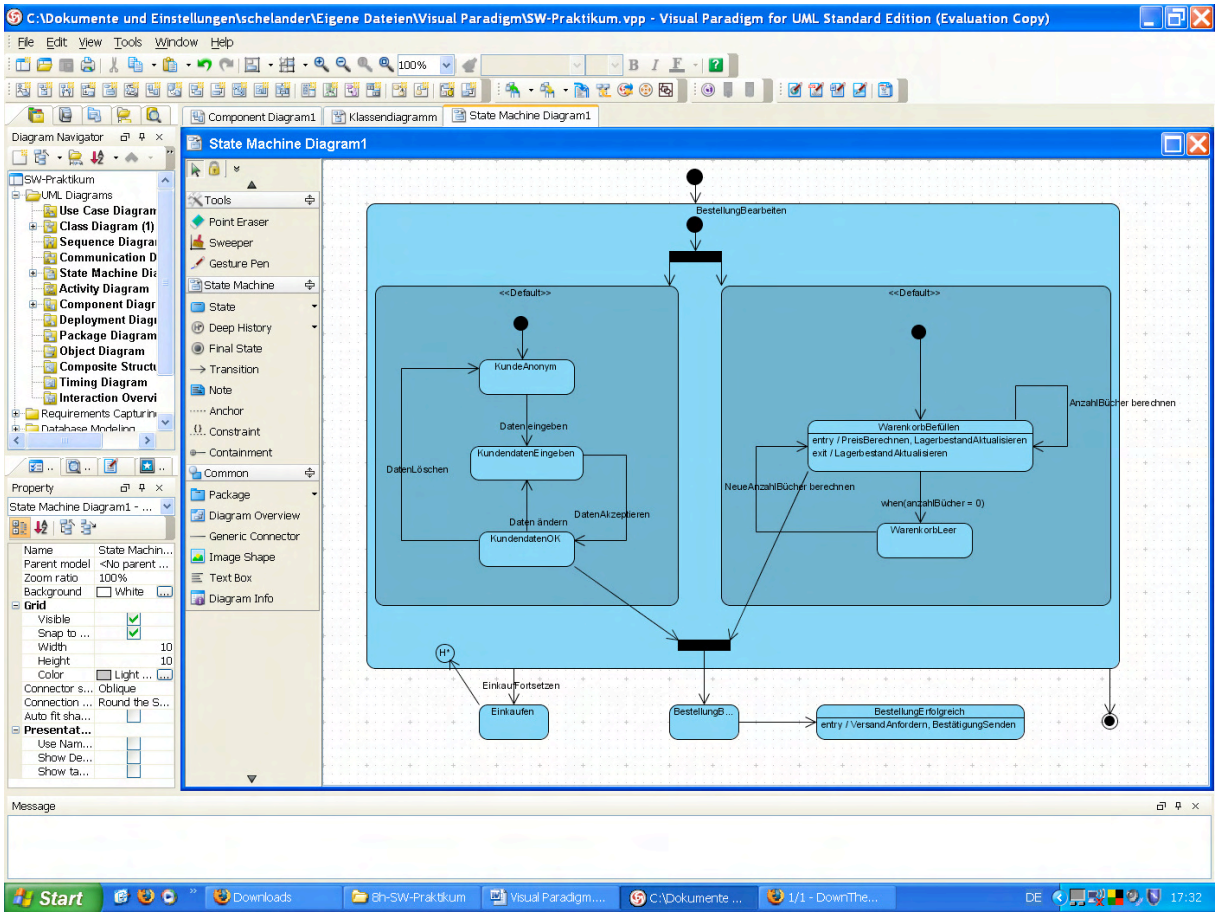
6.8.5. Zustandsdiagramm – Objecteering



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Anlegen eines Zustandsdiagramms.	LK auf Icon *StateMachine*, LK auf neu erstellte StateMachine, LK auf Icon *Create a state diagram*.	Maus: 3 Tastatur: 0 Wechsel: 0
		3
02 - Hinzufügen der drei großen Zustände.	LK auf Icon *Create a State* und LK auf Zeichenebene. Eingabe der Bezeichnung. Hier werden keine Stereotypen angeboten.	Maus: 6 Tastatur: 1 Wechsel: 2
		9
03 - Hinzufügen des Zustandes *WarenkorbBefüllen*.	LK auf Create a *State*, LK in Zeichenebene. Hinzufügen der Bezeichnung.	Maus:11 Tastatur:13 Wechsel:7
		31
04 - Hinzufügen des Zustandes *WarenkorbLeer*.	LK auf *State*, LK in Zeichenbereich und bezeichnen des Zustandes.	Maus: 2 Tastatur: 1 Wechsel: 1
		4

Schritt	Vorgangsweise	Benötigte Aktionen
05 - Setzen des Anfangszustandes.	LK auf *Create an initial state* und LK auf Zeichenbereich.	Maus: 2 Tastatur: 0 Wechsel: 0 2
06 - Hinzufügen der Verbindungen.	LK auf *Create a transition* und ziehen der Verbindungen. Bezeichnungen hinzufügen.	Maus: 18 Tastatur: 4 Wechsel: 5 27
07 - Hinzufügen der Zustände *KundeAnonym, KundendatenEingeben, KundendatenOK*.	LK auf *Create a State*, LK in Zeichenbereich und bezeichnen des Zustandes.	Maus: 6 Tastatur: 6 Wechsel: 12 24
08 - Setzen des Anfangszustandes.	LK auf Create an *InitialState* und LK auf Zeichenbereich.	Maus: 2 Tastatur: 0 Wechsel: 0 2
09 - Hinzufügen der Transitionen.	LK auf *Create a transition* und ziehen der Verbindungen. Bezeichnungen hinzufügen.	Maus: 23 Tastatur: 8 Wechsel: 7 38
10 - Auftragen der beiden Synchronisationen.	LK auf *Create a fork* und LK auf Zeichenbereich.	Maus: 4 Tastatur: 0 Wechsel: 0 4
11 - Einfügen der Zustände *BestellungBezahlen, BestellungErfolgreich* und *Einkaufen*.	LK auf *Create a state*, LK in Zeichenbereich und bezeichnen des Zustandes. Anlegen der Entrys.	Maus: 12 Tastatur: 14 Wechsel: 9 35
12 - Setzen des restlichen Anfangszustände und des Endzustandes.	LK auf *Create an initial state* und LK in Zeichenbereich.	Maus: 8 Tastatur: 0 Wechsel: 0 8
13 - Hinzufügen der restlichen Verbindungen.	LK auf *Create a transition* und ziehen der Verbindungen.	Maus: 35 Tastatur: 2 Wechsel: 1 38
GESAMT		**225**

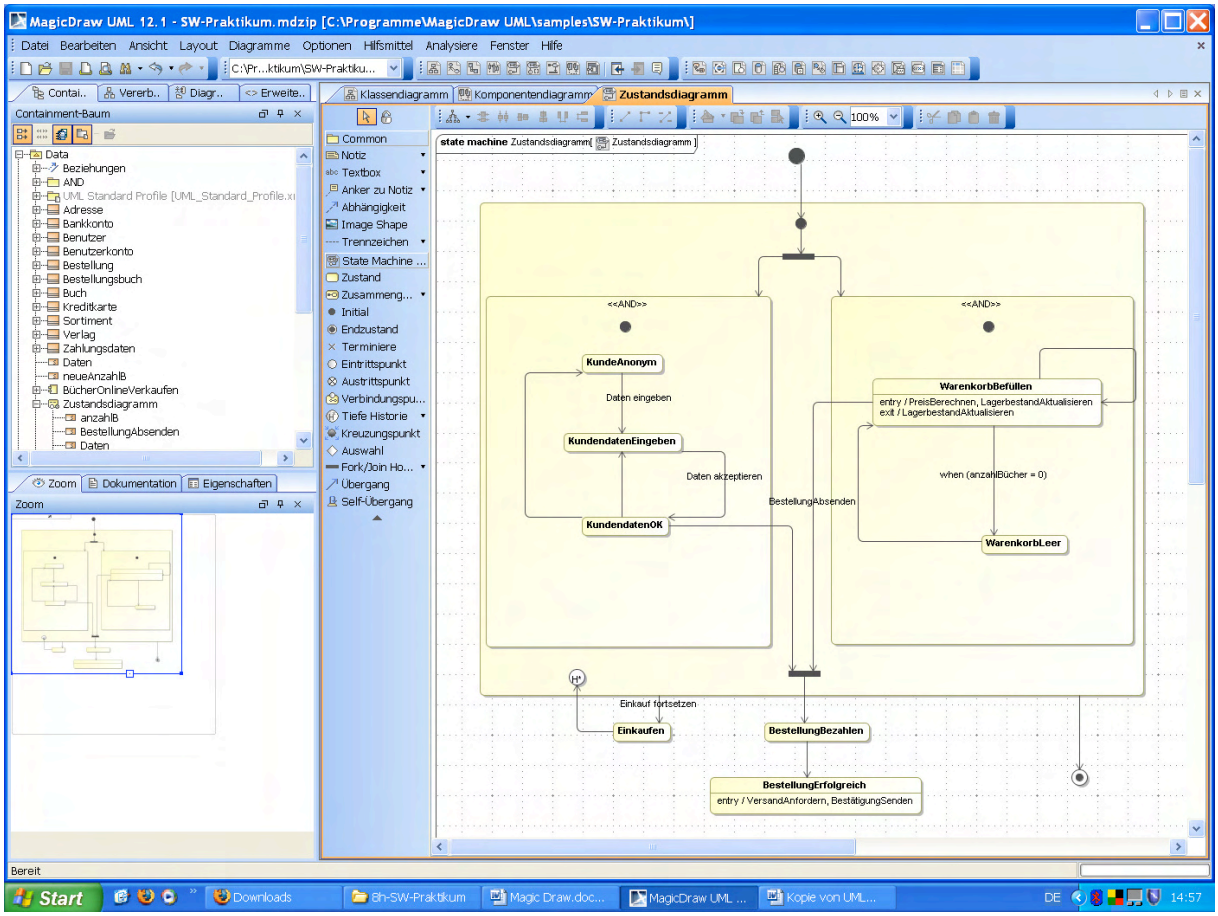
6.8.6. Zustandsdiagramm – Visual Paradigm



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Anlegen eines Zustandsdiagramms.	LK auf Icon *New State Machine Diagram*.	Maus: 1 Tastatur: 0 Wechsel: 0
		3
02 - Hinzufügen der drei großen Zustände.	LK auf Icon *State* und in die Zeichenebene ziehen. Eingabe der Bezeichnung und der Stereotypen. Löschen der Bezeichnungen der Subzustände	Maus: 14 Tastatur: 3 Wechsel: 5
		22
03 - Hinzufügen des Zustandes *WarenkorbBefüllen*.	LK auf *State* und in die Zeichenebene ziehen. Hinzufügen der Bezeichnung und des Inhalts.	Maus:6 Tastatur:6 Wechsel:6
		18
04 - Hinzufügen des Zustandes *WarenkorbLeer*.	LK auf *State*, in den Zeichenbereich ziehen. Bezeichung hinzufügen.	Maus: 1 Tastatur: 2 Wechsel: 1
		4

Schritt	Vorgangsweise	Benötigte Aktionen
05 - Setzen des Anfangszustandes.	LK auf *Initial Pseudo State* und in den Zeichenbereich ziehen.	Maus: 1 Tastatur: 0 Wechsel: 0 1
06 - Hinzufügen der Verbindungen.	LK auf *Transition* und ziehen der Verbindungen. Bezeichnungen hinzufügen.	Maus: 8 Tastatur: 3 Wechsel: 5 16
07 - Hinzufügen der Zustände *KundeAnonym, KundendatenEingeben, KundendatenOK*.	LK auf *State*, in den Zeichenbereich ziehen und Zustand benennen.	Maus: 3 Tastatur: 6 Wechsel: 5 14
08 - Setzen des Anfangszustandes.	LK auf *Initial Pseudo State* und in den Zeichenbereich.	Maus: 1 Tastatur: 0 Wechsel: 0 1
09 - Hinzufügen der Transitionen.	LK auf *Create a transition* und ziehen der Verbindungen. Bezeichnungen hinzufügen.	Maus: 10 Tastatur: 4 Wechsel: 7 21
10 - Auftragen der beiden Synchronisationen.	LK auf *Fork* bzw. *Join* und in den Zeichenbereich ziehen.	Maus: 2 Tastatur: 0 Wechsel: 0 2
11 - Einfügen der Zustände *BestellungBezahlen, BestellungErfolgreich* und *Einkaufen*.	LK auf *State*, in den Zeichenbereich ziehen. Bezeichnung hinzufügen. Anlegen der Entries.	Maus: 7 Tastatur: 4 Wechsel: 8 19
12 - Setzen des restlichen Anfangszustände und des Endzustandes.	LK auf *Initial Pseudo State, Final State* und *Deep History* und in den Zeichenbereich ziehen.	Maus: 3 Tastatur: 0 Wechsel: 0 3
13 - Hinzufügen der restlichen Verbindungen.	LK auf *Transition* und ziehen der Verbindungen.	Maus: 22 Tastatur: 1 Wechsel: 1 24
GESAMT		**148**

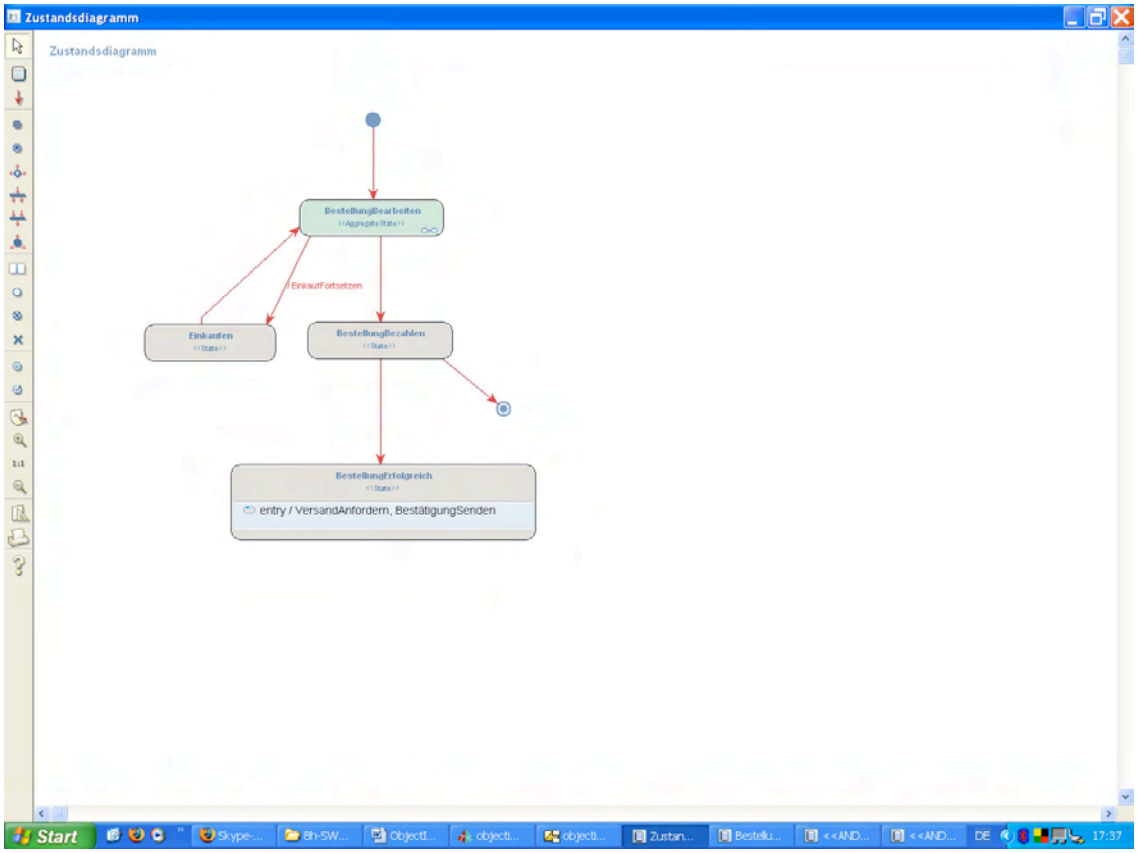
6.8.7. Zustandsdiagramm – Magic Draw

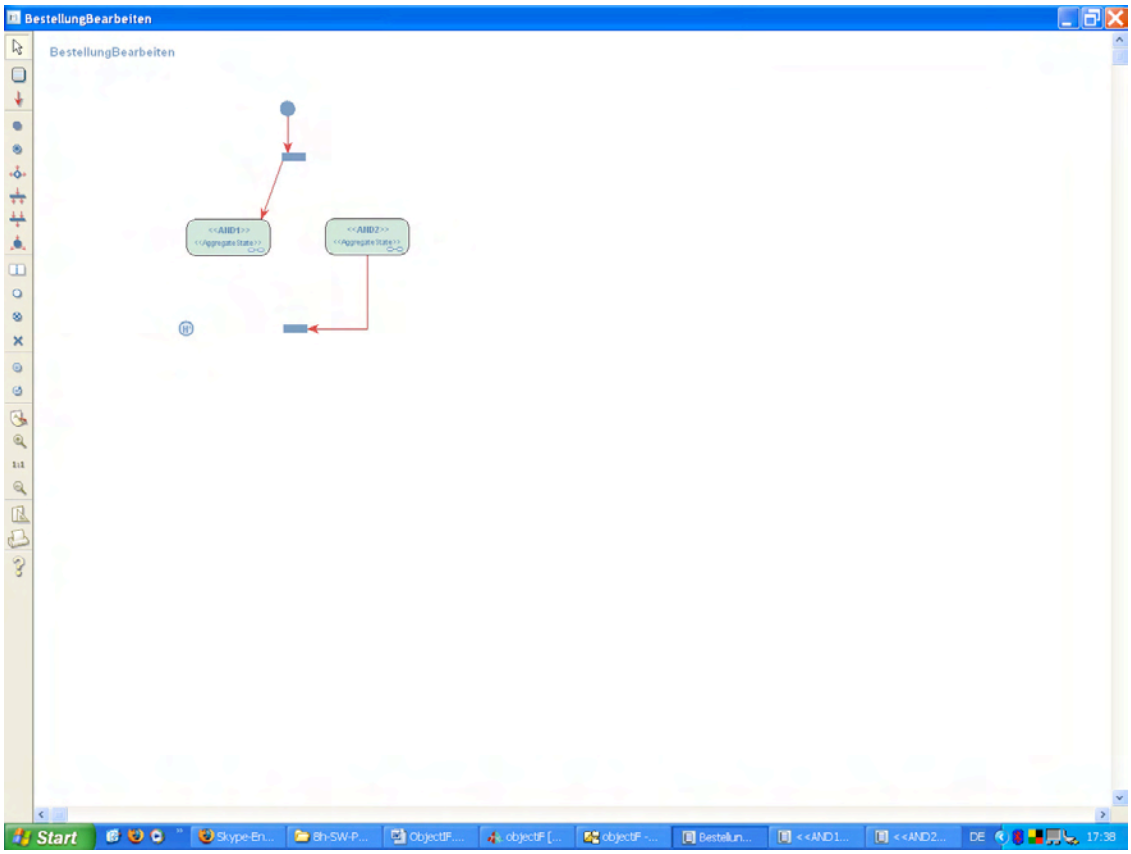


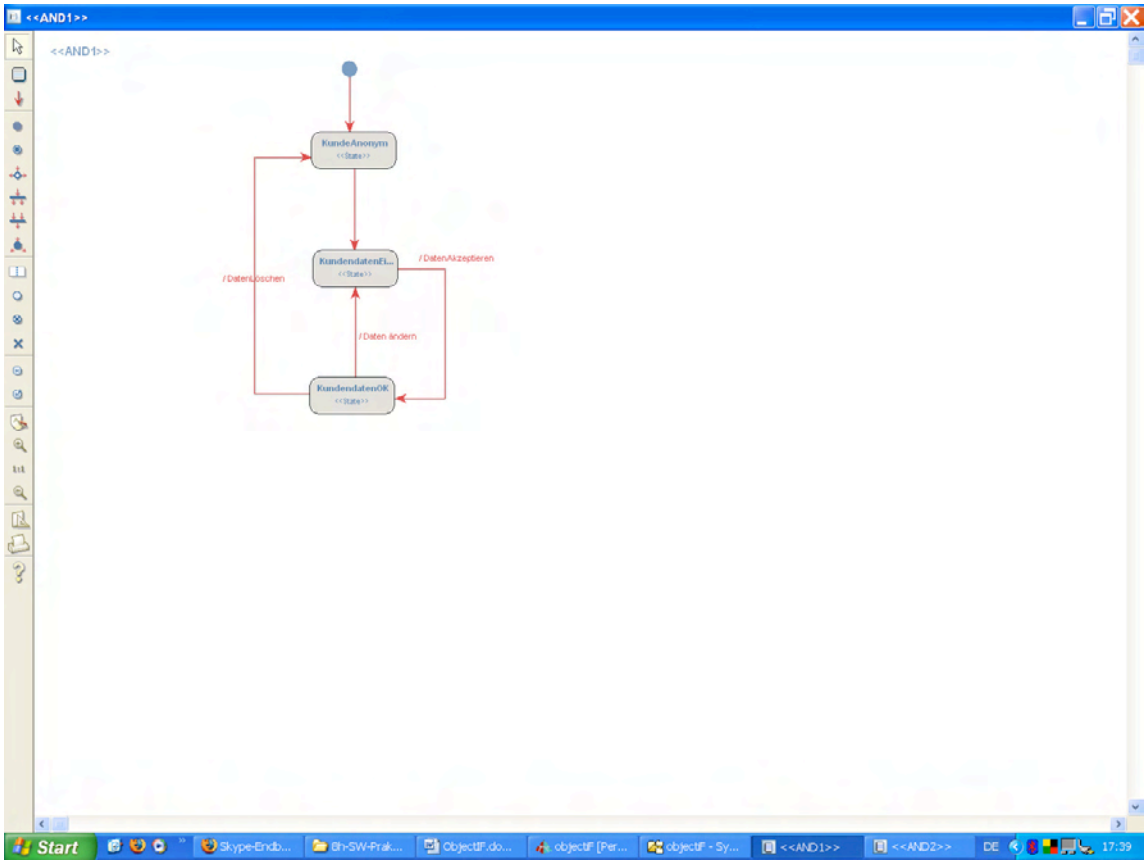
Schritt	Vorgangsweise	Benötigte Aktionen
01 - Anlegen eines Zustandsdiagramms.	LK auf Icon *Zustandsdiagramm*.	Maus: 1 Tastatur: 2 Wechsel: 1
		4
02 - Hinzufügen der drei großen Zustände.	LK auf *Zusammengesetzter Zustand* und in der Zeichenebene ziehen. Eingabe der Bezeichnung. Stereotypen über das Kontextmenü hinzufügen.	Maus: 12 Tastatur: 1 Wechsel: 2
		15
03 - Hinzufügen des Zustandes *WarenkorbBefüllen*.	LK auf *Zustand* und in die Zeichenebene klicken. Hinzufügen der Bezeichnung. Hinzufügen des Inhalts mittels DK auf den Zustand.	Maus:10 Tastatur:3 Wechsel:6
		19
04 - Hinzufügen des Zustandes *WarenkorbLeer*.	LK auf *Zustand.* LK im Zeichenbereich. Bezeichnung hinzufügen.	Maus: 2 Tastatur: 1 Wechsel: 1
		4

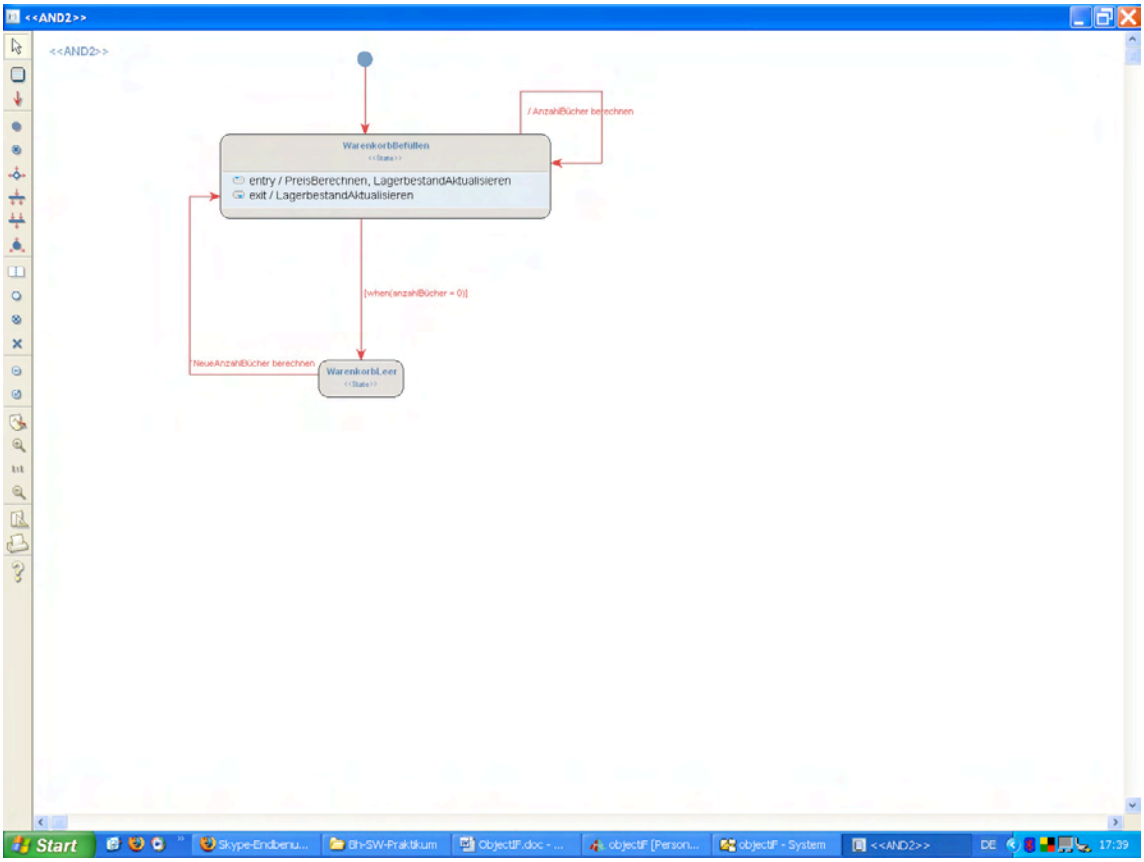
Schritt	Vorgangsweise	Benötigte Aktionen
05 - Setzen des Anfangszustandes.	LK auf *Kreuzungspunkt* und in den Zeichenbereich ziehen.	Maus: 2 Tastatur: 0 Wechsel: 0 2
06 - Hinzufügen der Verbindungen.	LK auf *Übergang* und ziehen der Verbindungen. Bezeichnungen hinzufügen.	Maus: 8 Tastatur: 3 Wechsel: 5 16
07 - Hinzufügen der Zustände *KundeAnonym, KundendatenEingeben, KundendatenOK*.	LK auf *Zustand.* LK im Zeichenbereich. Zustand beschriften.	Maus: 6 Tastatur: 3 Wechsel: 5 14
08 - Setzen des Anfangszustandes.	LK auf *Kreuzungspunkt* und in den Zeichenbereich.	Maus: 2 Tastatur: 0 Wechsel: 0 2
09 - Hinzufügen der Transitionen.	LK auf *Übergang* und ziehen der Verbindungen. Bezeichnungen hinzufügen.	Maus: 10 Tastatur: 4 Wechsel: 7 21
10 - Auftragen der beiden Synchronisationen.	LK auf *Fork/Join* und in den Zeichenbereich ziehen.	Maus: 4 Tastatur: 0 Wechsel: 0 4
11 - Einfügen der Zustände *BestellungBezahlen, BestellungErfolgreich* und *Einkaufen*.	LK auf *Zustand*, LK im Zeichenbereich. Bezeichnung hinzufügen. Anlegen der Entries.	Maus: 14 Tastatur: 5 Wechsel: 10 29
12 - Setzen des restlichen Anfangszustände und des Endzustandes.	LK auf *Initial, Kreuzungspunkt, Endzustand* und *Tiefe Historie* und LK im Zeichenbereich.	Maus: 8 Tastatur: 0 Wechsel: 0 8
13 - Hinzufügen der restlichen Verbindungen.	LK auf *Übergang* und ziehen der Verbindungen.	Maus: 22 Tastatur: 2 Wechsel: 3 27
GESAMT		**165**

6.8.8. Zustandsdiagramm – ObjectIF






<<AND1>>
KundeAnonym
KundendatenEi...
KundendatenOK
/ DatenAkzeptieren
/ DatenLöschen
/ Daten ändern

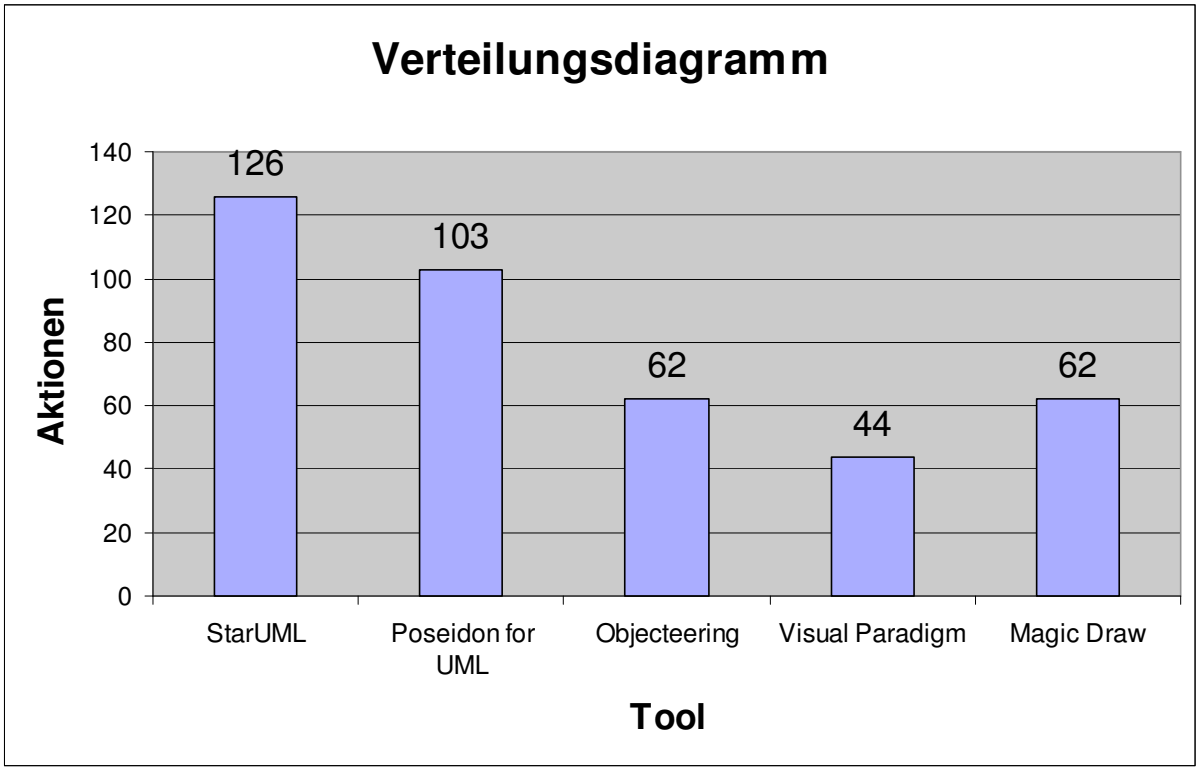

<<AND2>>
/ AnzahlBücher berechnen
WarenkorbBefüllen
entry / PreisBerechnen, LagerbestandAktualisieren
exit / LagerbestandAktualisieren
[when(anzahlBücher = 0)]
NeueAnzahlBücher berechnen
WarenkorbLeer

Schritt	Vorgangsweise	Benötigte Aktionen
01 - Anlegen eines Zustandsdiagramms.	Designansicht auswählen. Kontextmenü mittels RK aufrufen, LK auf *Zustandsdiagramm*, Name vergeben und bestätigen. Diagramm mittels DK öffnen.	Maus: 4 Tastatur: 2 Wechsel: 2 8
02 - Hinzufügen der drei großen Zustände.	LK auf Icon *Zustand*. LK in den Zeichenbereich. Name vergeben und bestätigen. Esc drücken, um wieder einen Mauszeiger zu bekommen. RK in den Zustand, LK auf *Anlegen, Subzustandsdiagramm*. Beide Subzustände anlegen.	Maus: 8 Tastatur: 8 Wechsel: 5 21
03 - Hinzufügen des Zustandes *WarenkorbBefüllen*.	Subzustandsdiagramm anlegen. Zustand zeichnen. Einträge vornehmen. Zustand vergrößern, damit die Einträge sichtbar werden.	Maus:8 Tastatur:4 Wechsel:6 18
04 - Hinzufügen des Zustandes *WarenkorbLeer*.	LK auf *State*, LK in den Zeichenbereich. Bezeichung hinzufügen und bestätigen.	Maus: 2 Tastatur: 2 Wechsel: 1 5
05 - Setzen des Anfangszustandes.	LK auf *Startzustand und LK* in den Zeichenbereich.	Maus: 2 Tastatur: 0 Wechsel: 0 2
06 - Hinzufügen der Verbindungen.	LK auf *Transition,* LK auf den ersten zu verbindenden Zustand, LK auf den zweiten Zustand. Bezeichnungen über RK und LK auf *Eigenschaften* hinzufügen.	Maus: 20 Tastatur: 6 Wechsel: 5 31
07 - Hinzufügen der Zustände *KundeAnonym, KundendatenEingeben, KundendatenOK*.	LK auf *Zustand*, LK im Zeichenbereich, Zustand benennen und bestätigen.	Maus: 6 Tastatur: 6 Wechsel: 5 17
08 - Setzen des Anfangszustandes.	LK auf *Startzustand* und in den Zeichenbereich.	Maus: 2 Tastatur: 0 Wechsel: 0 2
09 - Hinzufügen der Transitionen.	LK auf *Transition*, LK auf ersten Zustand, LK auf zweiten Zustand. Bezeichnungen hinzufügen.	Maus: 24 Tastatur: 6 Wechsel: 5 35
10 - Auftragen der beiden Synchronisationen.	LK auf *BestellungBearbeiten* in der Taskleiste. LK auf *Gabelung* bzw. *Vereinigung* und LK in den Zeichenbereich.	Maus: 5 Tastatur: 0 Wechsel: 0 5

Schritt	Vorgangsweise	Benötigte Aktionen
11 - Einfügen der Zustände *BestellungBezahlen, BestellungErfolgreich* und *Einkaufen*.	LK auf *Zustand*, LK im Zeichenbereich. Bezeichnung hinzufügen. Anlegen der Entries.	Maus: 7 Tastatur: 7 Wechsel: 8 22
12 - Setzen des restlichen Anfangszustände und des Endzustandes.	LK auf *Startzustand, Endzustand* und *Tiefe Historie* und LK im Zeichenbereich. Dazwischen über die Taskleiste zwischen den Subdiagrammen wechseln.	Maus: 9 Tastatur: 0 Wechsel: 0 9
13 - Hinzufügen der restlichen Verbindungen.	LK auf *Transition* und LK auf die zu verbinden Elemente. Bei der Gabelung wird nur eine der verzweigten Linien angezeigt, bei der Vereinigung nur eine der zusammengeführten Linien.	Maus: 28 Tastatur: 2 Wechsel: 2 32
GESAMT		**207**

6.9 Erstellung des Verteilungsdiagramms

6.9.1. Zusammenfassung der Ergebnisse

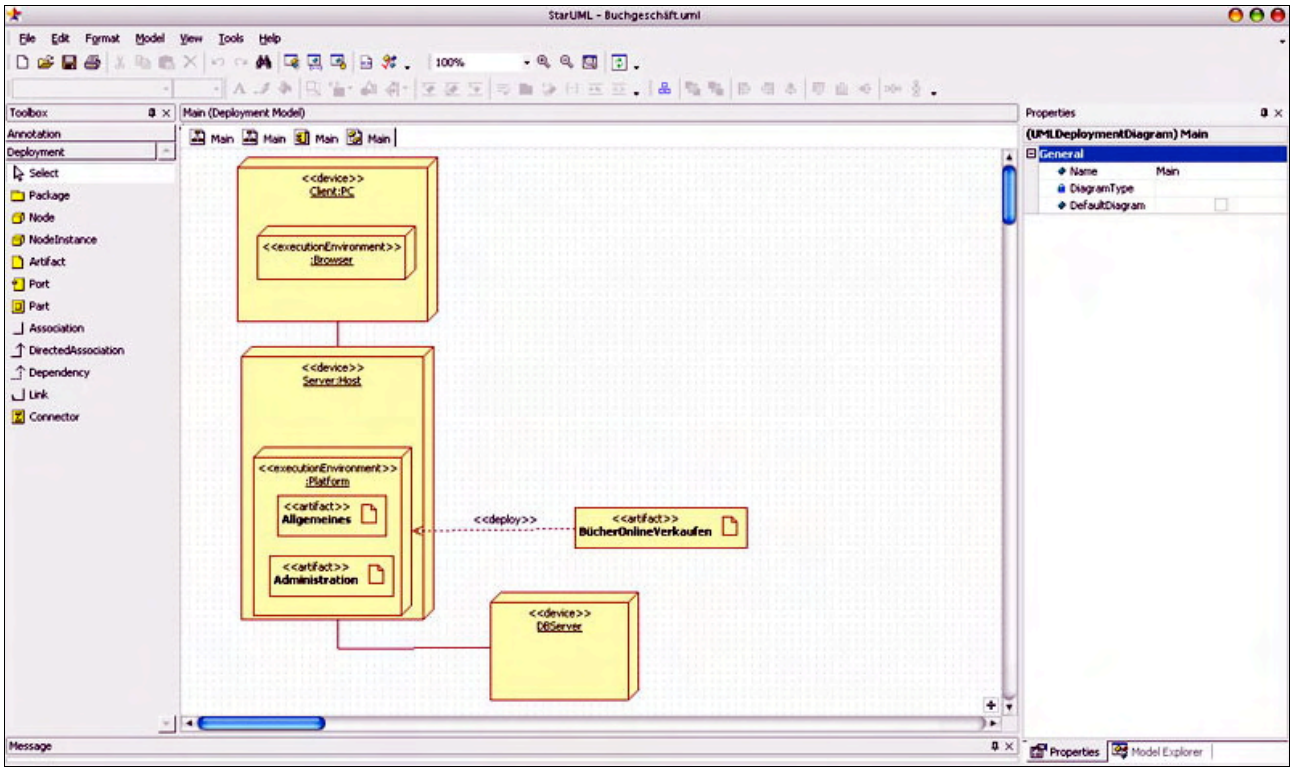


6.9.2. Interpretation der Ergebnisse

Bei der Erstellung dieser Diagrammart gibt es große Unterschiede zwischen den verschiedenen Werkzeugen im Bezug auf die benötigten Aktionen. StarUML und Poseidon for UML sind im Vergleich zu den anderen Tools weit abgeschlagen. Vergleicht man die einzelnen Aktivitäten so fällt auf, dass diese beiden Tools vor allem bei der Erstellung der Artefakte überdurchschnittlich viele Aktionen benötigen. Poseidon for UML benötigt hierfür 48 Aktionen, Visual Paradigm hingegen nur 15 Aktionen. In Visual Paradigm kann man Artefakte einfach durch Anklicken in der Werkzeugleiste und durch Ziehen in den Zeichenbereich erstellen.

Das Anlegen der Stereotype für die Geräte ist in StarUML umständlich, da man hierzu in das Eigenschaftsfenster wechseln muss und die Stereotype manuell eintragen muss. Die Erstellung eines Verteilungsdiagramms wird in ObjectIF nicht unterstützt.

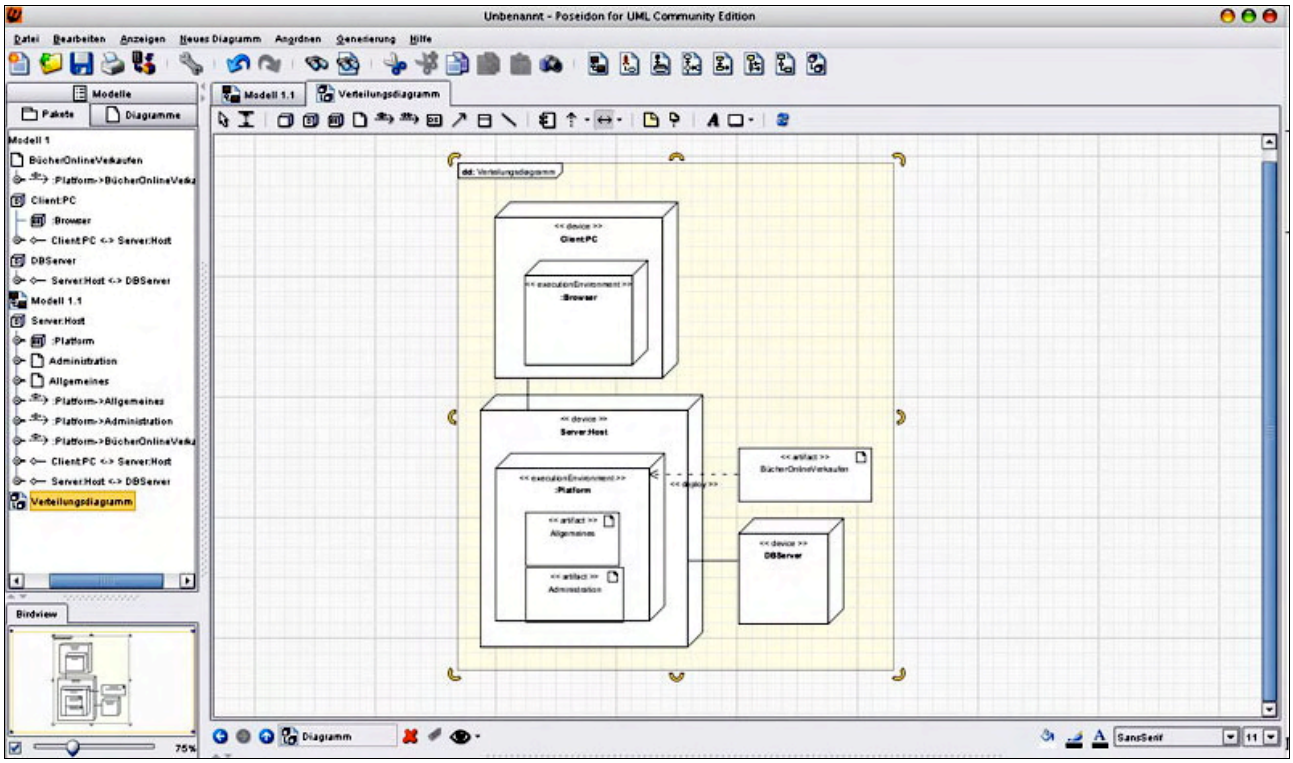
6.9.3. Verteilungsdiagramm – StarUML



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Neues Verteilungsdiagramm anlegen.	LK im Modellexplorer auf *Deployment Diagramms*.	Maus: 2 Tastatur: 0 Wechsel: 0 2
02 - Anlegen des Knotens *Client* und der Bedienungsumgebung, sowie setzen des Stereotypen.	LK für Werkzeugauswahl in Werkzeugleiste, unter Eigenschaften Eingabe des Stereotyps.	Maus: 10 Tastatur: 6 Wechsel: 12 28
03 - Anlegen des Knotens *Host* und der Bedienungsumgebung, sowie setzen des Stereotypen.	Analog Schritt 02.	Maus: 10 Tastatur: 6 Wechsel: 12 28
04 - Anlegen des Knotens *DBServer*, sowie Setzen des Stereotypen.	Analog Schritt 02.	Maus: 10 Tastatur: 6 Wechsel: 12 28
05 - Anlegen der Verbindungen zwischen Knoten.	LK auf Werkzeug *Link* und Ziehen der Verbindungen.	Maus: 4 Tastatur: 0 Wechsel: 0 4

Schritt	Vorgangsweise	Benötigte Aktionen
06 - Anlegen der drei Artefakte.	LK auf *Artifact* und LK in Zeichenbereich. Bestimmen der Bezeichnungen.	Maus: 6 Tastatur: 6 Wechsel: 12
		24
07 - Hinzufügen der Abhängigkeits-beziehung.	LK auf *Deploy* und Anlegen der Beziehung. Setzen des Stereotypen.	Maus: 6 Tastatur: 2 Wechsel: 4
		12
GESAMT		**126**

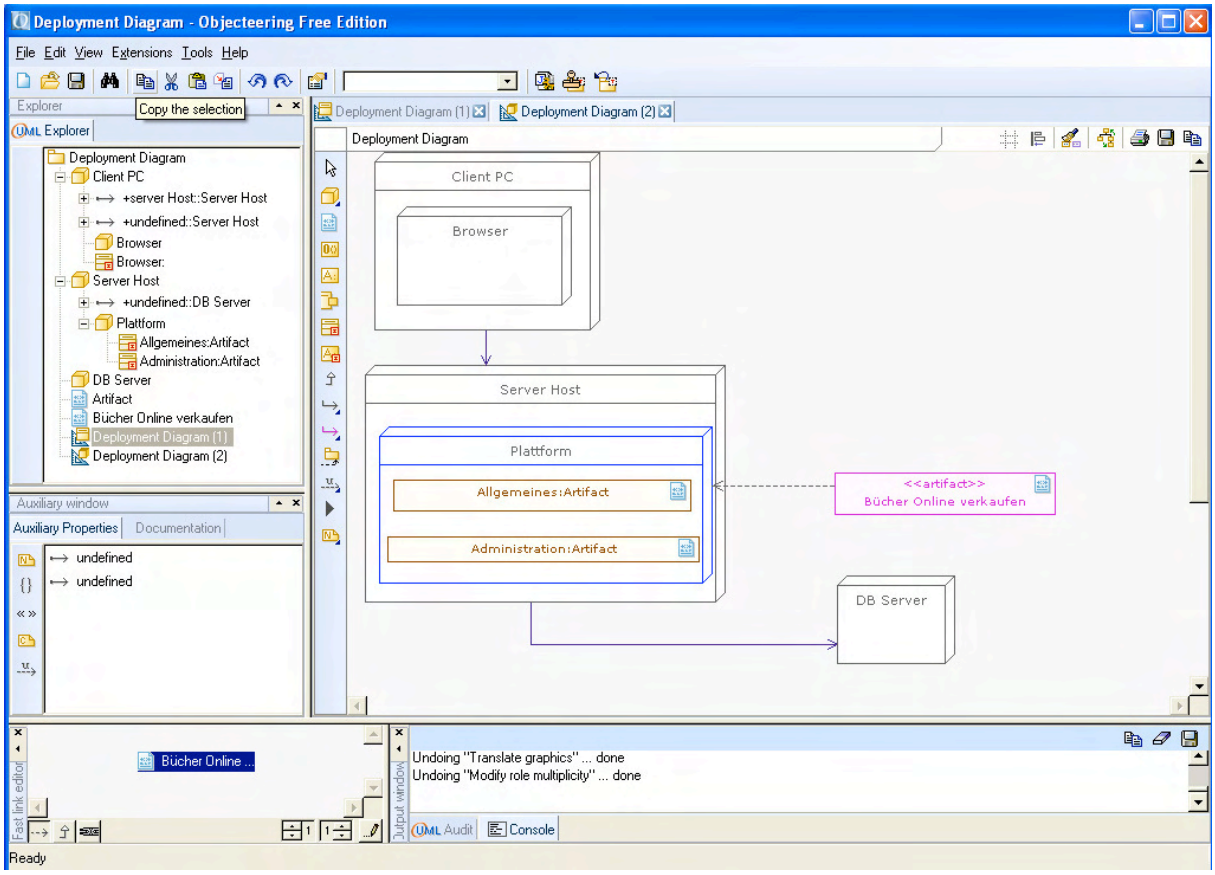
6.9.4. Verteilungsdiagramm – Poseidon for UML



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Neues Verteilungs-diagramm anlegen.	LK auf Icon *Verteilungsdiagramm* in Werkzeugleiste. Eingabe der Be-zeichnung.	Maus: 2 Tastatur: 2 Wechsel: 4
		8
02 - Anlegen des Gerä-tes *Client*.	LK auf Icon *Gerät hinzufügen*. Eingabe der Bezeichnung.	Maus: 2 Tastatur: 2 Wechsel: 4
		8
03 - Anlegen des Gerätes Server.	Analog Schritt 02.	Maus: 2 Tastatur: 2 Wechsel: 4
		8
04 - Anlegen des Gerä-tes *DBServer*.	Analog Schritt 02.	Maus: 2 Tastatur: 2 Wechsel: 4
		8
05 - Anlegen der beiden Ausführungsumge-bungen.	LK auf Icon *Ausführungsumgebung hinzufügen* und LK auf Zeichen-ebene. Angabe der Bezeichnung.	Maus: 4 Tastatur: 4 Wechsel: 8
		16

Schritt	Vorgangsweise	Benötigte Aktionen
06 - Hinzufügen der Artefakte	LK auf Artefakt-Icon und LK in Zeichenbereich. Benennen der Artefakte.	Maus: 12 Tastatur: 12 Wechsel: 24
		48
07 - Neue Verteilung hinzufügen.	LK auf *deploy*-Symbol und Auftragen der Verbindung.	Maus: 2 Tastatur: 0 Wechsel: 0
		2
08 - Verbinden der einzelnen Geräte mit Assoziationen.	LK in Drop-Down Menü und Auswahl der Assoziationsbeziehung. Verbinden der Geräte.	Maus: 5 Tastatur: 0 Wechsel: 0
		5
GESAMT		**103**

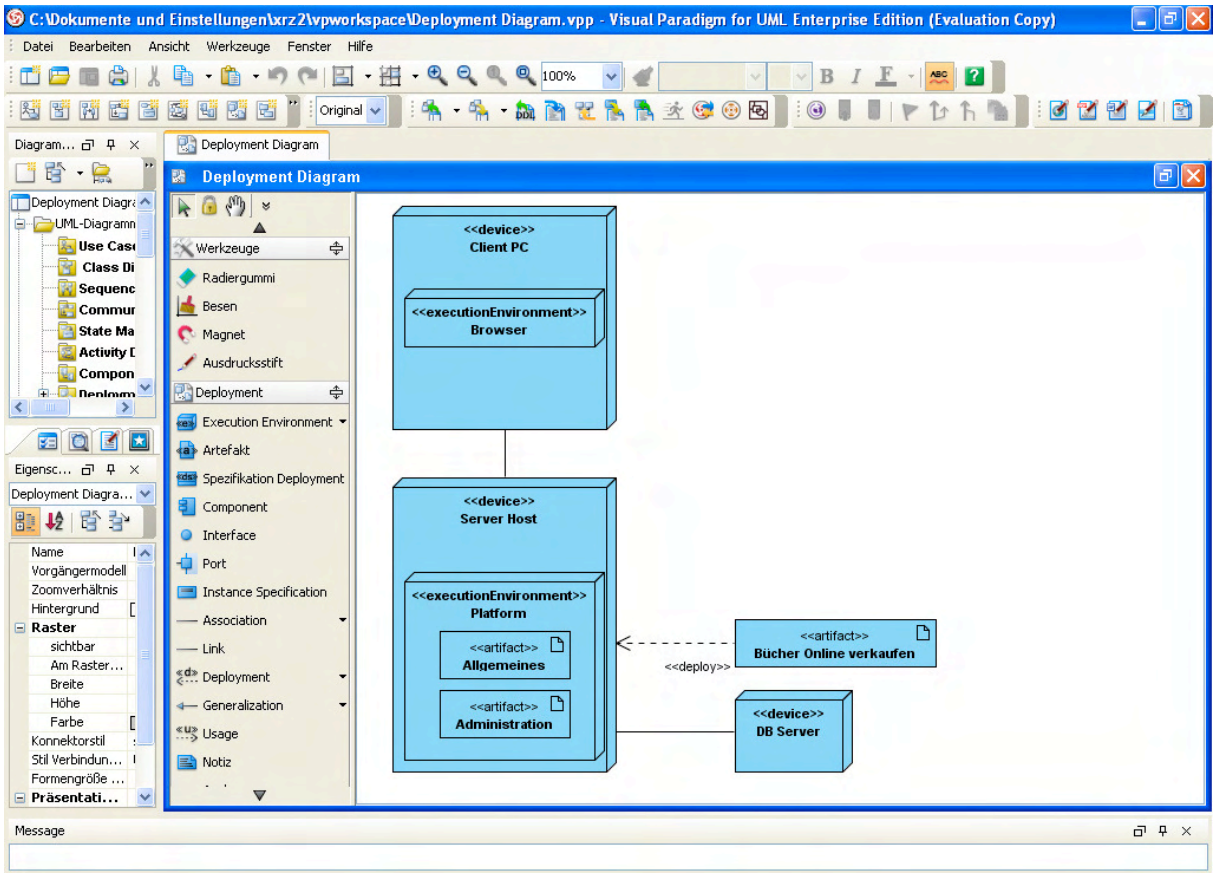
6.9.5. Verteilungsdiagramm – Objecteering



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Neues Verteilungs-diagramm anlegen.	LK im *UML Explorer* auf *Create a deployment diagram* und anschlie-ßend LK auf das UML Project um diese zu benennen.	Maus: 3 Tastatur: 1 Wechsel: 2
		6
02 - Anlegen des Gerä-tes *Client*.	LK auf Icon *Create a node*, LK auf die gewünschte Position und Eingabe der Bezeichnung.	Maus: 2 Tastatur: 1 Wechsel: 2
		5
03 - Anlegen des Gerätes Server.	Analog zu Schritt 02.	Maus: 2 Tastatur: 1 Wechsel: 2
		5
04 - Anlegen des Gerä-tes *DBServer*.	Analog zu Schritt 02.	Maus: 2 Tastatur: 1 Wechsel: 2
		5
05 - Anlegen der beiden Ausführungsumge-bungen.	Analog zu Schritt 02.	Maus: 4 Tastatur: 2 Wechsel: 4
		10

Schritt	Vorgangsweise	Benötigte Aktionen
06 - Hinzufügen der Artefakte	LK auf Artefakt-Icon und LK in Zeichenbereich. Benennen der Artefakte.	Maus: 13 Tastatur: 3 Wechsel: 6
		22
07 - Neue Verteilung hinzufügen.	LK auf *deploy*-Symbol, LK auf das Artefakt und LK auf die Zielposition.	Maus: 3 Tastatur: 0 Wechsel: 0
		3
08 - Verbinden der einzelnen Geräte mit Assoziationen.	LK create an association und LK auf die Geräteknoten.	Maus: 6 Tastatur: 0 Wechsel: 0
		6
GESAMT		**62**

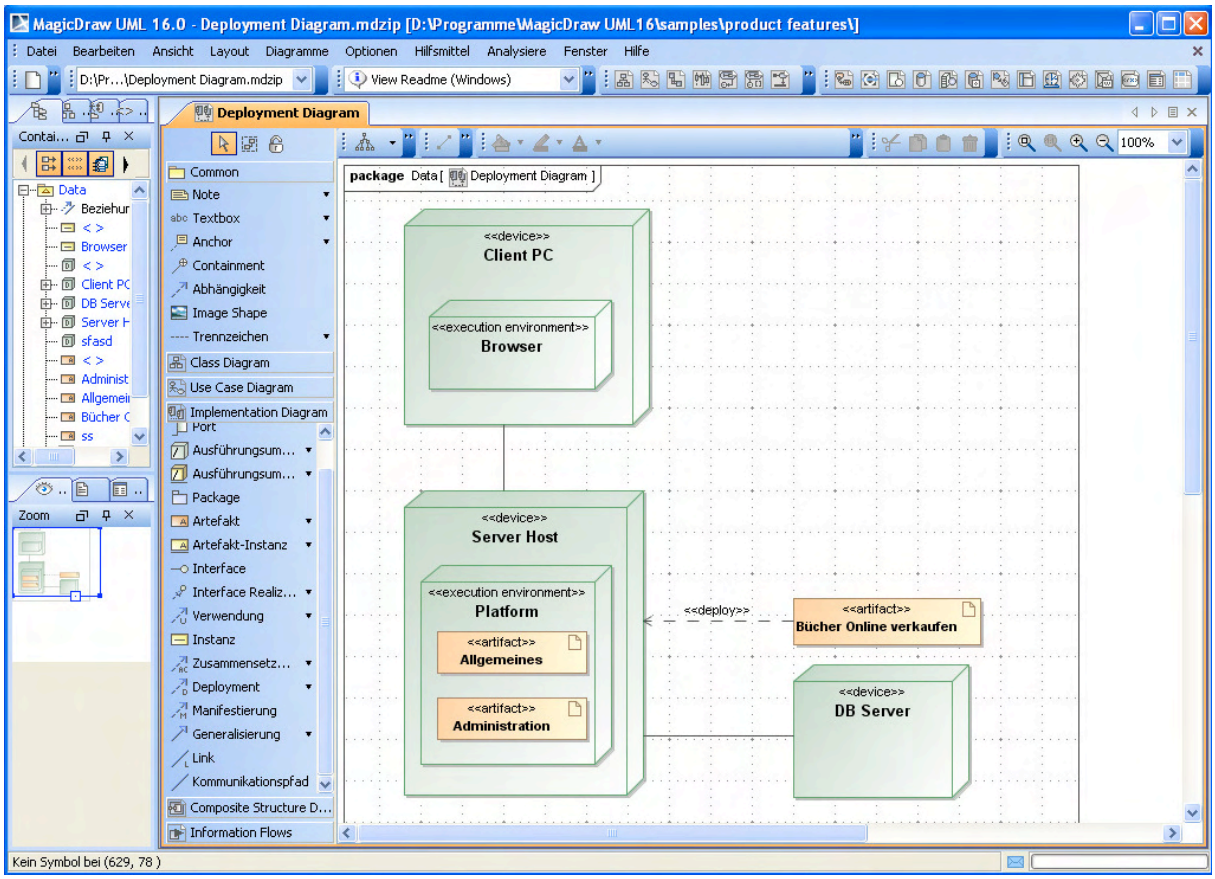
6.9.6. Verteilungsdiagramm – Visual Paradigm



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Neues Verteilungs-diagramm anlegen.	LK auf Icon *New Component Diagram*.	Maus: 1 Tastatur: 0 Wechsel: 0
		1
02 - Anlegen des Gerätes *Client*.	LK auf Icon *Device Node,* ziehen auf die gewünschte Position und Eingabe der Bezeichnung.	Maus: 2 Tastatur: 1 Wechsel: 2
		5
03 - Anlegen des Gerätes Server.	Analog zu Schritt 02.	Maus: 2 Tastatur: 1 Wechsel: 2
		5
04 - Anlegen des Gerätes *DBServer*.	Analog zu Schritt 02.	Maus: 2 Tastatur: 1 Wechsel: 2
		5
05 - Anlegen der beiden Ausführungsumgebungen.	LK auf Icon *Execution Environment* und ziehen auf die Zeichenebene. Eingabe der Bezeichnung.	Maus: 4 Tastatur: 2 Wechsel: 4
		10

Schritt	Vorgangsweise	Benötigte Aktionen
06 - Hinzufügen der Artefakte	LK auf Artefakt-Icon und ziehen in den Zeichenbereich. Benennen der Artefakte.	Maus: 6 Tastatur: 3 Wechsel: 6
		15
07 - Neue Verteilung hinzufügen.	LK auf *Deployment* und ziehen der Verbindung.	Maus: 2 Tastatur: 0 Wechsel: 0
		2
08 - Verbinden der einzelnen Geräte mit Assoziationen.	LK auf Association. Verbinden der Geräte.	Maus: 6 Tastatur: 0 Wechsel: 0
		6
GESAMT		**44**

6.9.7. Verteilungsdiagramm – Magic Draw



Schritt	Vorgangsweise	Benötigte Aktionen
01 - Neues Verteilungs-diagramm anlegen.	LK auf Icon *Implementationsdi-agramme*. Diagramm benennen und bestätigen.	Maus: 1 Tastatur: 2 Wechsel: 2
		5
02 - Anlegen des Gerä-tes *Client*.	LK auf Icon *Gerät*. LK auf die Gewünschte Position und Eingabe der Bezeichnung.	Maus: 3 Tastatur: 1 Wechsel: 2
		6
03 - Anlegen des Gerätes Server.	Analog zu Schritt 02.	Maus: 3 Tastatur: 1 Wechsel: 2
		6
04 - Anlegen des Gerä-tes *DBServer*.	Analog zu Schritt 02.	Maus: 3 Tastatur: 1 Wechsel: 2
		6
05 - Anlegen der beiden Ausführungsumge-bungen.	LK auf Icon *Ausführungsumgebung* und LK auf Zeichenebene. Angabe der Bezeichnung.	Maus: 6 Tastatur: 2 Wechsel: 4
		12

Schritt	Vorgangsweise	Benötigte Aktionen
06 - Hinzufügen der Artefakte	LK auf Artefakt-Icon und LK in Zeichenbereich. Benennen der Artefakte.	Maus: 9 Tastatur: 3 Wechsel: 6
		18
07 - Neue Verteilung hinzufügen.	LK auf *Deployment* und LK in den Zeichenbereich.	Maus: 3 Tastatur: 0 Wechsel: 0
		3
08 - Verbinden der einzelnen Geräte mit Assoziationen.	LK auf Kommunikationspfad und LK in den Zeichenbereich.	Maus: 6 Tastatur: 0 Wechsel: 0
		6
GESAMT		**62**

7 Gegenüberstellung der UML Werkzeuge

In der folgenden Tabelle werden alle sechs untersuchten UML-Werkzeuge anhand der im Abschnitt 5 vorgenommenen Bewertung gegenübergestellt.

Die Bewertung erfolgt gemäß folgendem Schema:

++	Alle Kriterien werden erfüllt
+	Die meisten Kriterien werden erfüllt
o	Kriterien werden mittelmäßig erfüllt
-	Großteil der Kriterien wird nicht erfüllt
- -	Keine der untersuchten Kriterien wird erfüllt

Methodenbasierte Kriterien	**StarUML**	**Poseidon for UML**	**Objecteering**	**Visual Paradigm**	**Magic Draw**	**ObjectIF**
Modellierung der Objektstruktur - Objektstatik	++	++	+	++	++	++
Modellierung der Systemstruktur Systemstatik	+	+	+	+	++	+
Dynamisches Objektverhalten - Objektdynamik	+	+	o	+	++	o
Zusammenhang zwischen Modellen	+	o	o	+	o	- -
Abstraktionsebenen / Sichten	o	o	+	+	++	-
Vorgangsreihenfolge bei der Entwicklung	++	++	o	+	++	++
Unterstützung bei der Methodenanwendung	o	o	+	o	+	o
Funktionale Kriterien	**StarUML**	**Poseidon for UML**	**Objecteering**	**Visual Paradigm**	**Magic Draw**	**ObjectIF**
Datenhaltung	++	+	+	++	++	+
Browsing-Eigenschaften	-	-	-	o	++	o
Navigation im Systemmodell	+	o	+	+	++	o
Konsistenzprüfung / Integrität	o	-	-	o	+	++
Report-Ausgabe	-	- -	- -	o	+	- -
Datenbankschema-Ausgabe	- -	- -	- -	o	++	- -
Diagramm-Ausgabe	o	o	-	+	+	++

Offenheit-Import/Export	o	o	-	+	-	-
Codegenerierung	+	o	+	+	+	+
Reverse-Engineering	o	++	++	++	++	+
Test / Verifikation	- -	- -	- -	- -	- -	o
Maßzahlen und Statistiken	- -	- -	- -	- -	++	-
Multi-User-Eigenschaften	-	-	- -	o	++	o
Security	-	-	-	o	o	o
Multi-Projekt-Eigenschaf-ten/Versionsmanagement	-	-	-	o	-	o
Ergonomie	**StarUML**	**Poseidon for UML**	**Objec-teering**	**Visual Para-digm**	**Magic Draw**	**ObjectIF**
Benutzeroberfläche	++	+	+	++	++	o
Tool-Dokumentation	o	o	o	o	+	+
Usability	+	o	-	o	+	-
Hilfefunktionen	++	o	-	+	+	o
Graphikeditor / Darstellung	+	o	o	+	+	-
Graphikeditor / Bedienung	+	+	o	+	+	+
Performance	++	o	++	o	+	+
Adaptivität	-	- -	- -	- -	o	- -

Interpretation des Vergleichs:

Es gibt keine großen Unterschiede bei den methodenbasierten Kriterien zwischen den sechs Tools. Mit allen Werkzeugen lassen sich die verschiedenen Diagramme einfach anlegen. Modellierungsinformationen werden zentral in einem Repository gespeichert. Einzig ObjectIF ist in der Anwendung wenig intuitiv. Auch ist es mit diesem Tool schwer möglich Zusammenhänge zwischen verschiedenen Diagrammtypen herzustellen. Die OCL-Notation wird von Objecteering, Visual Paradigm und Magic Draw unterstützt. Mit Ausnahme von ObjectIF kann auch das Abstraktionsniveau eingestellt werden. Am besten schneidet für diesen Bereich Magic Draw ab, gefolgt von Visual Paradigm.

Auch bei der funktionalen Bewertung hat Magic Draw die Nase vorn. Die leichte und übersichtliche Navigation sticht besonders hervor. Darüber hinaus ist es mit Magic Draw einfach möglich automatisch Reports zu generieren und Diagramme in Datenbankschemas zu transformieren. Ähnlich gut schneidet wiederum Visual Paradigm ab. Das Tool verfügt so wie Magic Draw über die meisten Funktionalitäten dieses Bereichs und ist auch von der Navigation annähernd gleich.

Gefolgt werden die zwei Spitzenreiter von StarUML, Poseidon for UML, Objecteering und ObjectIF. Diese vier Tools schneiden in diesem Bewertungsbereich fast gleich ab. StarUML hat seine Vorteile in der Navigation und den Browsing-Eigenschaften. Ein Nachteil von Poseidon for UML ist das Nichtvorhandensein einer Konsistenz- bzw. Integritätsprüfung. Alle vier Tools schneiden bei der Report-Ausgabe schlecht ab. ObjectIF punktet wiederum bei der Diagrammausgabe und ermöglicht wie auch Visual Paradigm und Magic Draw eine Mehrbenutzerverwaltung. Hierfür können eigene Server installiert werden.

Bei der Ergonomie schneiden StarUML, Visual Paradigm und Magic Draw am besten ab. Durch die integrierte und ausführliche Hilfefunktion dieser Tools kann dem Benutzer bei Problemen schnell geholfen werden. Das Arbeiten mit diesen Tools kann einfach und intuitiv erlernt werden. Auch Poseidon for UML besitzt eine Hilfe, jedoch nur im HTML-Format. Eine Suche nach Themengebieten ist nicht möglich.

Objecteering ist beim Anlegen von Diagrammen leicht verwirrend. Das Anlegen erfolgt mittels Rechtsklick auf das jeweilige Symbol. Die Bedienung von Poseidon for UML ist gewöhnungsbedürftig. In Poseidon for UML existieren zwei Möglichkeiten Diagramme zu erstellen. Die einzelnen Elemente können per Maus auf die Arbeitsfläche hinzugefügt werden. Die zweite Möglichkeit ist das Zeichnen von Diagrammen, ohne immer wieder das gewünschte Werkzeug auszuwählen. Hierfür werden die zur Verfügung stehenden Werkzeuge in Form von Icons auf die Zeichenfläche eingeblendet. Diese Art der Modellierung ist zwar schneller, benötigt aber viel Übung.

ObjectIF ist durch die Verwaltung mittels eigenständiger Fenster schwer überschaubar. Die Handhabung des Modelleditors hingegen ist leicht verständlich. Außerdem bietet ObjectIF wie die drei Spitzenreiter eine Online-Hilfe.

Abhängig von den gebotenen Funktionalitäten kann es bei der Performance zu leichten Verzögerungen kommen. Performanceprobleme sind beim Testen von Poseidon for UML einige Male aufgetreten. Hat der Benutzer mehrere Diagramme auf einmal geöffnet, so können geringe Wartezeiten auf ein Systemfeedback auftreten. Vor allem beim Laden von Projekten kann es bei allen Tools zu Wartezeiten kommen. Bei

der Performance schneiden StarUML und Objecteering am besten ab. Dies ist wahrscheinlich darauf zurück zu führen, dass die Spitzenreiter der vorherigen Kategorien, Visual Paradigm und Magic Draw, vom Funktionsumfang vielleicht etwas überladen sind.

Zusammenfassend sind Magic Draw und Visual Paradigm die Sieger der Bewertung. Mit diesen Tools werden alle wichtigen Funktionalitäten für die Modellierung mit der UML, das automatische generieren von Code und das Reverse Engineering geliefert. Der Umgang mit den Tools ist leicht erlernbar und intuitiv.

ObjectIF fällt vor allem durch die mangelnde Benutzerfreundlichkeit und die fehlende Möglichkeit Anpassungen zu machen auf. Dieses Tool bedarf einer etwas längeren Einarbeitungszeit und würde sich aus unserer Sicht für den Einsatz in Rahmen einer Lehrveranstaltung wenig eignen.

8 Zusammenfassung

In diesem Abschnitt werden die sechs zu untersuchenden UML-Werkzeuge anhand der wichtigsten Kriterien gegenüber gestellt.

Unterstützte Diagrammarten:

Symbole zur Bewertung:

+	Darstellung entspricht dem UML 2.0 Standard
-	Darstellung entspricht kaum oder nicht der UML
n	Konstrukt kann nicht dargestellt werden

Diagrammtyp	Star-UML	Poseidon for UML	Objec-teering	Visual Pardigm	Magic Draw	ObjectIF
Klassendiagramm	+	+	+	+	+	+
Objektdiagramm	+	+	+	+	+	+
Paketdiagramm	-	-	-	+	-	+
Komponentendiagramm	+	+	+	+	+	n
Kompositionsstrukturdiagramm	+	+	-	+	+	n
Verteilungsdiagramm	+	+	+	+	+	n
Anwendungsfalldiagramm	+	+	+	+	+	+
Aktivitätsdiagramm	+	+	+	+	+	+
Zustandsdiagramm	+	+	+	+	+	+
Sequenzdiagramm	+	+	+	+	+	+
Kommunikationsdiagramm	+	-	-	+	+	n
Zeitdiagramm	-	-	-	+	+	n

Datenhaltung:

Tool	Datenbasis	Browsing/Navigation	Konsistenzprüfung	Import/Export
StarUML	Filebasiert	Beides gut unterstützt, Windows Look and Feel	Konsistenzprüfung während der Modellierung und durch explizite Überprüffunktion	Import: XMI, Rational Rose Formate Export: XMI
Poseidon for UML	Filebasiert	Beides gut unterstützt, Look and Feel typischer Java Anwendungen	Keine Konsistenzprüfung	Import: XMI, ZUML, ZARGO Export: XMI, Grafikformate
Objecteering	Filebasiert	Mehrere Suchfunktionen, Look and Feel mittelmäßig	Implizite Konsistenzprüfung	Export: Model-Bus
Visual Paradigm	Filebasiert, Serverunterstützung	Beides gut unterstützt, Look and Feel	Implizite Konsistenzprüfung bzw. Auto-Korrektur	Import: XML, XMI, RoseProject, ERwinProject (XML) Export: XMI, XML, Bild

Tool	Datenbasis	Browsing/Navigation	Konsistenzprüfung	Import/Export
Magic Draw	Filebasiert	Beides gut unterstützt, Look and Feel	Implizite Konsistenzprüfung bzw. Auto-Korrektur	Import: MDXML, MDZIP, XML, XML.ZIP, XMI Export: XMI, MOF
ObjectIF	Filebasiert	Beides schlecht umgesetzt	Implizite Konsistenzprüfung	Import: .UML Export: .UML

Ein- und Ausgabemöglichkeiten:

Tool	Reports	Ausgabeformate für Reports	Drucken
StarUML	Notwendiges Add-In kostenlos verfügbar	doc, xls, txt, html	Windows-konforme Druckerschnittstelle
Poseidon for UML	Keine Möglichkeit Reports zu erstellen	-	Windows-konforme Druckerschnittstelle
Objecteering	Keine Reports	-	Windows-konforme Druckerschnittstelle
Visual Paradigm	Report-Erstellung integriert; zusätzlich kann ein Berichts-Generator heruntergeladen werden	html, pdf, word	Windows-konforme Druckerschnitts
Magic Draw	Report-Erstellung integriert	html, pdf, rtf	Windows-konforme Druckerschnitts
ObjectIF	Keine Reports	-	Windows-konforme Druckerschnittstelle

Code Generierung und Reverse Engineering:

Tool	Zielsprachen	Reverse Engineering	DB-Schema-Generierung
StarUML	C++, C#, Java	Sourcecode kann eingelesen werden auch wenn dieser nicht von StarUML generiert wurde	Nicht möglich
Poseidon for UML	Java	Sourcecode kann eingelesen werden auch wenn dieser nicht von Poseidon for UML generiert wurde	Nicht möglich
Objecteering	Java, C++, Fortran	Sourcecode in Java, C++ und C# kann unabhängig vom generierenden Tool eingelesen werden.	Nicht möglich

Tool	Zielsprachen	Reverse Engineering	DB-Schema-Generierung
Visual Paradigm	Java, C#, VB.NET, PHP, ODL, ActionScript, C++, IDL, Delphi, Perl, XSD, Python, Objectice-C, Ada95, Ruby	Java, C++, .NET dll oder exe, CORBA IDL, Ada 9x, XML, JDBC, Hibernate, PHP 5.0, Python, Objective-C	Nur mit Zusatzmodul möglich
Magic Draw	Java, EJB, C#, C++, CORBA IDL, DDL, WSDL und XML Schema	Java, C#, C++, CORBA IDL, EJB 2.0, DDL, CIL (MSIL), WSDL, und XML Schema	der Modell Transformation-Assistenten kann Datenbankschemas für relationale und objektorientierte Datenbanken erzeugen
ObjectIF	Java, C#, C++, BPEL, XSD und WSDL	Java, C#, C++, .Net Projektmappen, BPEL, XSD und WSDL	Nicht möglich

Mehrbenutzerkriterien:

Tool	Netzwerkfähigkeit	Koordination/ Synchronisation	Security	Multi-Projekt-Verwaltung
StarUML	Windowssysteme	Keine E-Mail/ Talk-Mechanismen	Benutzerverwaltung und Logging nicht vorhanden	Keine Versionsverwaltung; kopieren von Modellen zwischen Projekten möglich
Poseidon for UML	Betriebssystemunabhängig	Keine E-Mail/ Talk-Mechanismen	Benutzerverwaltung und Logging nicht vorhanden	Keine Versionsverwaltung
Objecteering	Windows, Linux, Solaris	Keine E-Mail/ Talk-Mechanismen	Benutzerverwaltung und Logging nicht vorhanden	Keine Versionsverwaltung
Visual Paradigm	Java, Linux, Mac, Windows	Keine E-Mail/ Talk-Mechanismen	Benutzerverwaltung am Server möglich	Versionsverwaltung und Konfliktmanagement am Server
Magic Draw	Java 5 oder 6 kompatible VM. Windows NT/2000/XP/Vista, Linux, Mac OS X	Keine E-Mail/ Talk-Mechanismen	Benutzerverwaltung am Server möglich	Keine Versionsverwaltung
ObjectIF	Windowssysteme	Keine E-Mail/ Talk-Mechanismen	Benutzerverwaltung am Server möglich	Keine Versionsverwaltung

Benutzerfläche und Support:

Tool	Darstellung	Dokumentation und Hilfesystem	Usability	Adaptivität
StarUML	Windows-konforme Darstellung, Look and Feel	Hilfefunktion integriert, UML-Beispiele vorhanden, Onlinehilfe	Tool in englisch und koreanisch verfügbar, Alignmentfunktion, genaue Fehlermeldungen	Symbolleisten an Benutzer anpassbar, Bildschirmelemente anpassbar
Poseidon for UML	Typische Java Anwendung, Look and Feel	UML-Beispiele vorhanden, Onlinehandbuch	Tool in deutsch und englisch verfügbar, genaue Fehlermeldungen	Größe der Bildschirmelemente anpassbar, Symbol- und Menüleisten können nicht verändert werden
Objecteering	Ähnlich Java-Anwendungen, look and feel mittelmäßig	Dokumentation wird mitinstalliert, keine Beispiele vorhanden	Tool nur in englisch verfügbar, Fehlermeldungen ohne Hilfestellung	Keine Adaptivität.
Visual Paradigm	Windows-konforme Darstellung, look and feel	Dokumentation kann mitinstalliert werden, keine Beispiele vorhanden	Tool in englisch, deutsch, japanisch und chinesich verfügbar, wenige Fehlermeldungen	Praktisch nicht vorhanden
Magic Draw	Windowskonforme Standarddarstellung, kann durch den Benutzer angepasst werden, look and feel gut umgesetzt	Hilfefunktion integriert, UML-Beispiele vorhanden, Onlinehilfe, Benutzerhandbuch in englischer Sprache	Tool in deutsch und englisch verfügbar, jedoch vermischen beider sprachen bei deutscher Oberfläche, genaue Fehlermeldungen	Größe der Bildschirmelemente anpassbar, Symbol- und Menüleisten können ebenfalls angepasst werden.
ObjectIF	Keine windows-konforme Darstellung, look and feel kaum umgesetzt	Hilfefunktion integriert, UML-Beispiele vorhanden, Onlinehilfe	Tool in deutsch und englisch verfügbar, genaue Fehlermeldungen	Symbolleisten nicht anpassbar

9 Anhang

9.1 Kriterienkatalog aus Kapitel 4

Methodenbasierte Kriterien

a) Modellierung der Objektstruktur-Objektstatik

1. Die Reihenfolge der Komponenten einer Klasse im Klassendiagramm kann der Benutzer bestimmen.
2. Für jedes Attribut kann ein Datentyp festgelegt werden.
3. Für eine Methode kann ein Rückgabetyp angegeben werden, falls eine Methode einen Wert bzw. Objekt zurückliefert.
4. Für eine Methode können Parameter angegeben werden, die beim Methodenaufruf übergeben werden. Dabei kann auch spezifiziert werden, ob es sich um Eingangs-, Ausgangs- oder Durchgangsparameter handelt.
5. Für eine Methode können Vor- und Nachbedingungen angegeben werden, die erfüllt sein müssen (Assertions).

b) Modellierung der Systemstruktur-Systemstatik

1. Auch rekursive Beziehungen können erfasst werden.
2. Einmal erfasste Beziehungen können jederzeit nachträglich verändert werden, sofern die Änderungen mit den bisher erfassten Inhalten verträglich sind.
3. Unabhängig von der Zuordnung von Klassen zu Packages lassen sich Klassen nach bestimmten Gesichtspunkten in Klassenkategorien einteilen.
4. Es werden OCL-Annotationen unterstützt.

c) Dynamisches Objektverhalten-Objektdynamik

1. Für jedes Objekt bzw. jede Klasse (Abstraktion) kann ein Zustandsdiagramm erstellt werden. Dieses ist umkehrbar eindeutig mir der jeweiligen Klasse verbunden.
2. Es können Bedingungen modelliert werden, die einen Zustandsübergang bewirken. Dabei ist ein Bezug auf Attribute eines Objekts möglich.
3. Das Werkzeug kann Aktionen darstellen, die einem Zustandsübergang zugeordnet sind. Dabei ist ein Bezug auf Methoden des Objekts möglich.
4. Das Tool ermöglicht die Verfeinerung von Aktivitäten und Ereignissen im Sinne einer genaueren Beschreibung.
5. Die Darstellung von geschachtelten Zustandsdiagrammen, die eine Generalisierung von Zuständen modellieren, ist möglich. Dabei muss ein Objekt in irgendeinem Zustand des untergeordneten Zustandsdiagramms (Unterstatus) sein.
6. Die Modellierung einer Generalisierung von Ereignissen ist möglich. Diese ist unabhängig von der modellierten Klassenhierarchie.
7. Bei Sequenzdiagrammen kann auf Methoden bzw. Attribute des statischen Objektmodells Bezug genommen werden.

d) Zusammenhang zwischen den Modellen

1. Jedes Zustandsdiagramm ist umkehrbar eindeutig mit einem Objekt bzw. einer Klasse verbunden.
2. Das dynamische Verhalten (Zustandsdiagramm) einer Klasse wird an ihre Unterklasse vererbt.
3. Ereignisse aus der Modellierung des dynamischen Objektverhaltens können Methoden der jeweiligen Klasse zugeordnet werden.
4. Bei der Modellierung von Zuständen im dynamischen Modell kann auf Attribute des Objektmodells Bezug genommen werden.

e) Abstraktionsebenen / Sichten

1. Der Detaillierungsgrad von Klassendiagrammen am Bildschirm lässt sich durch Ausblenden bzw. Einblenden von Attributen und Methoden individuell einstellen.
2. Die Darstellung von Attributen und Methoden ist auf unterschiedlichem Detaillierungsniveau möglich (Ein-/Ausblenden von Typen, Rückgabetypen, Parameter, Parametertypen etc.).
3. Die Angaben über die Sichtbarkeit von Objektkomponenten können für eine Variation der Genauigkeit der Darstellung herangezogen werden (z.B. lediglich Anzeige der öffentlichen Komponenten).
4. Das Detaillierungsniveau der Diagrammanzeige kann sowohl für jedes Diagramm einzeln, als auch generell als Default-Einstellung festgelegt werden.

f) Vorgangsreihenfolge bei der Entwicklung

1. Das Werkzeug zwingt zu einer bestimmten Vorgangsweise, was die Reihenfolge des Anlegens von Modellierungsinformationen betrifft. Klassen etwa, können zunächst nur mit einem Namen angelegt und erst später genauer beschrieben werden.
2. Modellelemente können direkt textuell im Data-Dictionary erfasst und erst später in ein oder mehrere Diagramme eingezeichnet werden.
3. Neu anzulegende Modellelemente können direkt in ein Diagramm eingezeichnet werden, wobei die entscheidenden Informationen automatisch sofort im Data-Dictionary erfasst werden.

g) Unterstützung bei der Methodenanwendung

1. Zur Unterstützung bei der Anwendung von UML lasen sich aus dem Tool Informationen zu deren Funktionen, Notation und Semantik in einer hypertextartigen „Benutzerhilfe“ ausgeben.
2. Die Notationselemente des Tools entsprechen 1:1 jenen der UML, so dass die Verwendung von Basisliteratur zur Methode keine Verwirrung auslöst.
3. Bei der Eingabe von Modellierungsinformationen, die der Konsistenzprüfung bzw. den Integritätsbedingungen nicht standhalten, erscheint eine Warnung und eine Erklärung der entsprechenden Konzepte ist abrufbar.

Funktionale Kriterien

a) Datenhaltung

1. Modellierungsinformationen, die in mehreren Diagrammen vorkommen, sind in der Basis nicht redundant bzw. nur einmal gespeichert (Integration).
2. Modellierungsinformationen, die in der Datenbasis erfasst sind, müssen nicht zwingend in einem Diagramm vorkommen.
3. Modellinformationen können auch direkt in der Datenbasis gewartet (insbesondere auch gelöscht) werden.
4. Das Datenformat kann als „offen“ bezeichnet werden und ist nicht ausschließlich in diesem Tool anzutreffen.
5. Der Import und Export von Daten über Schnittstellen ist möglich.

b) Browsing-Eigenschaften

1. Das Auffinden von beliebigen Elementen (Klassen, Attribute, Methoden, etc.) durch Eingabe ihres Namens ist möglich.
2. Elemente können auch nach Ähnlichkeitskriterien, durch Pattern Matching, gesucht werden.
3. Die Suche von Elementen kann durch bloße Eingabe der Anfangsbuchstaben vorgenommen werden, wodurch die jeweils zutreffenden Elemente zur Auswahl angeboten werden.
4. Bei der Suche können Einschränkungen vorgenommen werden, etwa durch das Setzten von Filtern.
5. Als Ergebnis einer Suche erhält man das gesuchte Element mit weiteren wichtigen Informationen – z.B. für eine Klasse: Oberklasse, Unterklassen, Beziehungen zu weiteren Klassen, Attribute, Methoden.
6. Die Angaben, die zu einem Modellierungselement gemacht werden, dienen unmittelbar als Link zum Zugriff auf genauere Informationen über diese selbst.

7. Das Werkzeug ermöglicht die Durchführung von Änderungen an beliebigen Modellierungsinformationen direkt aus dem Browser heraus. Diagramme, auf die solche Änderungen Auswirkungen haben, werden sofort automatisch angepasst.
8. Das Werkzeug ermöglicht auch in textueller Form eine übersichtliche Darstellung von Klassenhierarchien.
9. Das Tool ermöglicht auf einfache Art die Navigation innerhalb einer Klassenhierarchie (z.B. durch einfaches Anklicken der Oberklasse).
10. Es gibt die Möglichkeit, für jedes Modellierungselement eine Aufstellung zu erhalten, in welchen übergeordneten Strukturen es vorkommt (z.B. in welchen Klassen kommt das Attribut „GebDatum“ vor?).
11. Alle Modellelemente, die in der Datenbasis gespeichert sind, aber noch in keinem Diagramm vorkommen, können auf einfache weise ausgewählt werden.

c) Navigation im Systemmodell

1. Das Werkzeug erlaubt eine rasche und übersichtliche Navigation innerhalb bzw. zwischen den Darstellungen für einzelne Modellierungsaspekte eines Systems.
2. Es gibt die Möglichkeit, ausgehend von einem Diagramm direkt zur jeweils übergeordneten bzw. verfeinerten Darstellung zu wechseln. Z.B. kann durch einen einfachen Mausklick aus einem Zustandsdiagramm zum entsprechenden Klassendiagramm verzweigt werden.
3. Die einzelnen Elemente eines Diagramms sind nicht nur textuelle bzw. graphische Darstellungen sondern gleichzeitig Hot-Links zum Aufrufen von elementspezifischen, genaueren Informationen.

d) Konsistenzprüfung / Integrität

1. Es können keine Assoziationen, Datenflüsse oder Transaktionen angelegt werden, die ins „Leere“ gehen.
2. Rekursive Vererbungsstrukturen können nicht erstellt werden.
3. Mehrere Modellelemente mit gleichem Namen können in einem Paket nicht angelegt werden.
4. Beim Anlegen von Namen, die bereits bestehenden ähnlich sind, wird der Benutzer gewarnt. Z.B. „GebDatum“ und „Gebdatum“.
5. Das Tool verfügt über eine Funktion, mit der ein bestehendes Modell bzw. der gesamte Datenbestand in einer Datenbasis auf Konsistenz geprüft werden kann. Dabei werden auch Beziehungen zwischen den Teilmodellen bzw. zwischen Diagrammen geprüft.
6. Wird ein Modellierungselement aus einem Diagramm gelöscht, darf es nicht gleich automatisch aus der Datenbasis gelöscht werden (referentielle Integrität).

e) Report-Ausgabe

1. Basierend auf den Daten des Data-Dictionary lassen sich automatisch Auswertungen mit einstellbarem Detaillierungsgrad ausgeben.
2. Für die statische Objektstruktur (Klassen, Attribute, Methoden etc.) können geeignete Reports ausgegeben werden.
3. Für das dynamische Objektverhalten lassen sich geeignete Reports ausgeben,
4. Für die statische Systemstruktur (Vererbungs-, Aggregations-, Assoziationsbeziehungen) lassen sich geeignete Reports erstellen.
5. Auch zur Darstellung des dynamischen Systemverhaltens gibt es Reports (z.B. über message-passing).
6. Ein Report, der über die Ergebnisse einer expliziten Konsistenzprüfung Informationen enthält, lässt sich generieren.
7. Reports können auch einfach auf Grund von ad-hoc Abfragen oder auf andere, frei definierte Weise erstellt werden.
8. Die Ausgabe von Reports in verschiedenen Formaten ist möglich (PDF, PS, usw.).

f) Datenbankschema-Ausgabe

1. Die Generierung eines Datenbankschemas für relationale Datenbanken auf Basis der Informationen des Objektmodells ist möglich.
2. Die Generierung eines objektorientierten Datenbankschemas auf Basis der Informationen des Objektmodells ist möglich.
3. Die Angabe von Schlüsselattributen für eine spätere DB-Schema-Ausgabe ist möglich.

4. Zusätzlich zu den Datentypen für die bei der Codegenerierung zu verwendende Zielsprache können bei den Attributen auch Domains angegeben werden.
5. Alle datenbankrelevanten Informationen aus der Klassenstruktur und der statischen Struktur des Anwendungssystems finden sich in den generierten Datenbankschemata wieder (Attribute, Beziehungen, etc.).

g) Diagramm-Ausgabe

1. Der Diagrammausdruck kann als präsentationsfähig bezeichnet werden.
2. Alle Diagramme lassen sich so ausdrucken, dass sie durch einfaches Zusammenleben von A4 Blättern übersichtliche Dokumentationen ergeben.
3. Die gedruckten Diagramme entsprechen genau der Darstellung am Bildschirm.
4. Das Abstraktionsniveau lässt sich für die Diagrammausgabe frei wählen (z.B. nur Klassennamen anzeigen / Klassen, Attribute u. Methoden anzeigen etc.).
5. Bei der Diagrammausgabe sind die Ausrichtung (Quer- oder Hochformat) und Skalierung der Darstellung einstellbar.

h) Offenheit-Import / Export

1. Das Tool ist ein Modul aus der Produktpalette des Herstellers, welches mit Zusatzprodukten des Herstellers zusammenarbeitet und auf diese Weise modulartig erweitert werden kann.
2. Das Tool schöpft die von der Plattform gebotenen Möglichkeiten des Datenaustauschs sowohl für Text als auch für graphische Darstellungen aus (z.B. Cut&Paste von Grafiken und Texten über die Zwischenablage).
3. In den Graphikeditor können auch geläufige toolfremde Grafikformate eingelesen werden.
4. In den Zeicheneditoren können auch geläufige Textformate direkt eingelesen werden.
5. Es gibt bereits Zusatztools von Drittherstellern, die Ausgaben des Tools einlesen und weiterverarbeiten können.

i) Codegenerierung

1. Die automatische Codegenerierung für eine oder mehrere in Frage kommende objektorientierte Zielsprache ist möglich.
2. Für jede im Modell vorhandene Klasse wird im Sourcecode der Zielsprache eine entsprechende Codierung vorgenommen.
3. Attribute werden inklusive ihrer Typen in die Zielsprache umgesetzt.
4. Methoden werden inklusive ihrer Argumente (inkl. Typen) und Rückgabetypen in den entsprechenden Sourcecode richtig umgesetzt.
5. Beziehungen zwischen Klassen werden im resultierenden Sourcecode richtig umgesetzt.
6. Für Methoden werden Sourcecode-Gerüste produziert, die noch durch den Inhalt ergänzt werden müssen.
7. Zugriffsbeschränkungen werden auch im Programmcode berücksichtigt.
8. Strukturierungen oberhalb des Niveaus der Klassen finden sich in der physischen Struktur des Codes wieder (z.B. mehrere Module führen zu einzelnen Source-Dateien).
9. Das Ausmaß der Codegenerierung lässt sich durch Parameter bestimmen (z.B. Defaultwerte, Datentypen, Rückgabetypen, Argumente, Zugriffsbeschränkungen, Präfix für Klassennamen, etc.).
10. Wenn generierter Code händisch erweitert wurde und die automatische Codegenerierung erneut angestoßen wird, bleibt der eigene (user added) Code erhalten (z.B. Kennzeichnung durch toolseitig vergebenen Kommentaren).
11. Inhalte von Methoden lassen sich schon in der textuellen Beschreibung der Methode eintragen und werden in den Code übernommen.

j) Reverse-Engineering

1. Bestehender Programmcode kann mit dem Tool eingelesen werden. Dieses erzeugt aus den Code-Informationen die entsprechenden A/D-Diagramme.
2. Es kann auch Code eingelesen werden, der nicht mit dem Tool selbst generiert wurde.

3. Codefragmente, die über Klassendefinitionen, Komponenten und Beziehungen hinausgehen (z.B. der Inhalt von Methoden) können in einem Editor bearbeitet und erneut als Code gespeichert werden.

k) Test / Verifikation

1. Das Tool erlaubt die textuelle Beschreibung von Testfällen für einzelne Objekte bzw. Methoden
2. Das Tool ermöglichte eine Simulation auf Modellebene, bei der das Senden von Nachrichten simuliert und das entsprechende Objektverhalten dargestellt wird.

l) Maßzahlen und Statistiken

1. Das Tool kann auf Basis der Daten des Repository geeignete Software-Maßzahlen berechnen, etwa die Tiefe von Vererbungshierarchien oder Kopplungsmaßzahlen etc.
2. Die automatische Ausgabe von Statistiken (Anzahl modellierter Klassen, Attribute, Methoden, etc.) ist möglich.
3. Die Ausgabe von Informationen über den Projektstatus (Mitarbeiter, Zugriffsberechtigungen, letzte Änderungen etc.) ist möglich.

m) Multi-User-Eigenschaften

1. Das Tool kann von mehreren Benutzern gleichzeitig verwendet werden.
2. Die Benutzer des Werkzeugs können – unabhängig vom Basis-Betriebssystem – in User und Gruppen eingeteilt werden (tooleigenes „Userkonzept“).
3. Es gibt einen Konfliktlösungsmechanismus, der verhindert, dass kein Benutzer die Arbeit eines anderen versehentlich zerstört (Locking).
4. Der Umfang des Lockings ist einstellbar – z.B. von Modellebene bis zur Ebene eines Attributs.

n) Security

1. Jeder Benutzer muss sich mit seinem User-Namen und einem Passwort ausweisen, bevor er das Werkzeug nutzen kann (Login/Logout).
2. Zugriffsrechte für Benutzer können auf User- oder Gruppenebene festgelegt werden.
3. Über ein automatisch mitgeschriebenes Protokoll können Änderungen auch im Nachhinein auf dem dafür verantwortlichen Benutzer zurückgeführt werden (Logging).
4. Das Tool verfügt über eine geeignete Möglichkeit zur Datensicherung, die auch automatisierbar ist.
5. Für den Fall von Systemabstürzen verfügt das Werkzeug über einen geeigneten Recovery-Mechanismus, der die Integrität des Programms und der Daten gewährleistet.
6. Jeder Benutzer kann Elemente eines Modells vorübergehend vor Änderungen schützen (einfrieren), auch falls er nicht gerade daran arbeitet und eingeloggt ist.
7. Im Falle von Bedienungsfehlern verbleibt das Werkzeug in einem definierten Zustand und gibt eine Fehlermeldung aus.
8. Unbeabsichtigte Löschungen können durch „Bestätigungsabfragen“ verhindert werden.
9. Löschungen können bis einem Gewissen Grad automatisch rückgängig gemacht werden (generelle Undo-Funktion).

o) Multi-Projekt-Eigenschaften / Versionsmanagement

1. Das Werkzeug ermöglicht die zeitlich parallele Abwicklung von mehreren Projekten.
2. Der Austausch von Modellierungsinformationen zwischen einzelnen Projektbasen ist möglich.
3. Der Bezug auf bereits in anderen Projekten gespeicherten Modellierungsinformationen (Referenzen) ist möglich. Somit können bereits erstellte Modelle wieder verwendet werden, ohne diese kopieren zu müssen.
4. Das Werkzeug erlaubt die gleichzeitige Verwaltung von mehreren verschiedenen Versionen eines Systemmodells.
5. Das Tool kann automatisch Änderungen einer Version gegenüber der Ausgangsversion anzeigen.

Ergonomie

a) Benutzeroberfläche

1. Das allgemeine Erscheinungsbild ist optisch ansprechend und übersichtlich.
2. Das GUI ist so implementiert, dass das Look-and-Feel der entsprechenden Benutzeroberfläche vorhanden ist.
3. Dialoge und Eingabemasken sind übersichtlich und zweckmäßig zu bedienen.
4. Die Namensgebung von Menüaktionen und Auswahlmöglichkeiten sind weitgehend selbstbeschreibend, einheitlich und widerspruchsfrei.
5. Momentan nicht verfügbare Menübefehle oder Optionen sind besonders dargestellt.
6. Im Falle von länger dauernden Aktionen erhält der Benutzer Informationen über Status und Verarbeitungsfortschritt (z.B. Fortschrittsbalken, Sanduhr).
7. Die Anzeige von Listen am Bildschirm ist einheitlich und übersichtlich.

b) Tool-Dokumentation

1. Es gibt ein ausführliches, gedrucktes Handbuch zur Bedienung des Tools.
2. Es gibt schriftliche Anleitungen zur Installation.
3. Das Handbuch ist in der jeweiligen Landessprache (Deutsch) verfasst.
4. Es gibt ein Kurz-Handbuch für versierte Benutzer.
5. Es gibt ein „Erste-Schritte“ Handbuch für unerfahrene Benutzer.
6. Das Handbuch hat ein Inhaltsverzeichnis und einen Stichwortindex bzw. Referenzteil.
7. Das Handbuch lässt sich leicht ergänzen, Teile lassen sich leicht austauschen.
8. Es gibt ein hypertextartiges Onlinehandbuch.
9. Das Onlinehandbuch ist direkt aus dem Tool heraus abrufbar.
10. Das Onlinehandbuch kann im Hintergrund offen gehalten werden.
11. Mit dem Tool wurden fertige Beispiele geliefert, die als Anregung und zur Einführung verwendet werden können, und die sich zum Selbststudium eignen.
12. Wichtige Informationen, die sich erst nach Vorliegen der Dokumentation ergeben, sind in Textdateien vorhanden, auf die auch die Dokumentation verweist.

c) Usability

1. Der Abbruch von Aktionen ist zu jeder Zeit möglich.
2. Im Fehlerfall wird der Benutzer durch eine informative und weiterführende Fehlermeldung benachrichtigt. Er erhält auf Wunsch auch direkt Hilfe zur aktuellen Situation.
3. Für wichtige Menüfunktionen gibt es Abkürzungs-Tastaturkombinationen (Tastatur-Shortcuts).
4. Funktionstasten können mit oft benötigten Befehlsfolgen belegt werden.
5. Das Tool bietet für ein rasches Arbeiten Kontextmenüs mit den jeweils anwendbaren, wichtigsten Befehlen an.
6. Die Eingabe von ungültigen Werten bzw. Ausführung ungültiger Operationen führt zu einer kontrollierten Fehlerbehandlung: Es erscheint eine Fehlermeldung und der kontrollierte Programmablauf wird fortgesetzt.
7. Es gibt Symbolleisten (Toolbar, Buttonbar), aus denen man häufig benötigte Befehle sofort abrufen kann – etwa eine Werkzeugleiste mit Symbolen zur Objektmodellierung. Die Symbolleisten sind vom Benutzer individuell konfigurierbar.
8. In einer Statuszeile finden sich hilfsreiche Informationen – etwa über den gerade auszuführenden Befehl und die weitere Vorgangsweise.
9. Mehrfenstertechnik wird unterstützt d.h. mehrere Darstellungen desselben Projekts, z.B. verschiedene Diagramme können gleichzeitig nebeneinander betrachtet werden.
10. Die Selektion aus Listen ist auf einfache Weise möglich (Einfach- und Mehrfachauswahl).
11. Es gibt eine Version des Werkzeugs in deutscher und englischer Sprache.

d) Hilfefunktionen

1. Das Tool bietet eine kontextsensitive Hilfefunktion, die leicht erreichbar ist (z.B. F1).
2. Das Werkzeug hat eine Hypertexthilfe, welche alle Hilfethemen umfasst.
3. Die Toolhilfe ermöglicht das Suchen nach bestimmten Hilfethemen.
4. Das Werkzeug bietet eine Hot-Help an: Wird die Maus auf ein Element, bzw. einen Menübefehl bewegt, erscheint dazu auf Wunsch die Hilfe.

e) Graphikeditor / Darstellung

1. Die Symbole sind in einem bestimmten Umfang standardmäßig anpassbar (als Voreinstellung).
2. Zusätzlich zu den Diagrammen aus UML können „ freie“ Diagramme aus den gängigsten Umrissformen (Rechtecke, Ovale, Linien, etc.) mit Beschriftungen erstellt werden.
3. Die Darstellung der Diagramme am Bildschirm entspricht dem späteren Ausdruck (WYSIWIG).
4. Für Notationselemente können verschiedene, selbst definierbare Farben voreingestellt werden.
5. Es gibt eine Funktion zur horizontalen bzw. vertikalen Ausrichtung von Text oder Symbolen (Alignment).
6. Die Bildschirmdarstellung ganzer Diagramme oder Teile eines Diagramms kann (stufenlos) vergrößert und verkleinert werden (Zoom).
7. Es gibt eine Funktion zum Rotieren von Symbolen und vor allem von Texten.
8. Es existiert eine Funktion, die ein Diagramm automatisch in der optimalen Größe und ohne Überlappungen von Elementen am Schirm anzeigt.
9. Da Layout eines Diagramms, welches man speichert und wieder lädt, wird beibehalten.
10. Die Schriftgröße zur Beschreibung von Diagrammelementen ist frei wählbar und die Schrift selbst gut lesbar.
11. Die Größe aller Notationselemente ist einstellbar, aber insgesamt einheitlich und nicht etwa von der Länge der darin enthaltenen Beschreibung (z.B. Klassenname) abhängig.
12. Ein Zeilenumbruch innerhalb einer Elementbeschriftung ist möglich.
13. Für alle Linien (bei Assoziationen) können auch nachträglich beliebige Eckpunkte festgelegt werden, um Überschneidungen zu vermeiden.
14. Der Editor erlaubt die Einstellung von Strichstärke und Linienart.
15. Es gibt eine Funktion zum „Neuzeichnen“ des Bildschirminhaltes durch das System (Refresh).

f) Graphikeditor / Bedienung

1. Das Einzeichnen von Notationselementen in ein Diagramm erfolgt komfortabel durch einfaches Aufziehen mit der Maus oder durch Auswahl aus einem Kontextmenü.
2. Sofort nach dem Anlegen eines Notationselements kann dieses textuell beschrieben werden (z.B. Klassen-, Attribut- und Methodenname).
3. Eine Funktion zum vielfachen Einfügen desselben Elements hintereinander steht zur Verfügung (Tool halten).
4. Notationselemente können durch einfaches drag-and-drop an eine beliebige Stelle eines Diagramms verschoben werden.
5. Der Zeichenbereich für Diagramme ist „unendlich“ groß. Werden Diagramme größer als eine Bildschirmseite, kann horizontal und vertikal gescrollt werden.
6. Zum Zweck einer automatischen, horizontalen, bzw. vertikalen Ausrichtung von Diagrammelementen kann ein frei definierbares Raster angelegt werden (Grid Alignment).
7. Auf Wunsch erscheinen Linien und Symbole exakt dort, wo man sie haben will.
8. Man kann mehrere Symbole gruppieren (z.B. durch Umrahmen mit dem Maus-Cursor), um darauf eine Operation auszuführen (z.B. Verschieben).
9. Das Abbrechen einer Zeichenaktion (z.B. irrtümliches Aufziehen einer Assoziation zu einer noch nicht gezeichneten, zweiten Klasse) ist einfach durchführbar.
10. Es gibt eine Undo-Funktion, die den letzten Bearbeitungsschritt (oder mehrere) rückgängig macht.
11. Es gibt eine Redo-Funktion, die den letzten Bearbeitungsschritt wiederholt.
12. Die allgemeine Bedienung des Graphikeditors verdient das Prädikat „intuitiv“.

13. Im Graphikeditor wird das Kopieren/Ausschneiden und Einfügen von Symbolen unterstütz (Copy/Cut und Paste).
14. Symbole lassen sich leicht wieder aus einem Diagramm entfernen, etwa durch Markieren und Betätigen der „Enft“-Taste.

g) Performance

1. Bei der Darstellung der Diagramme und deren Manipulation (Zoomen, Scrollen, etc.) erfolgt der Bildaufbau in Anbetracht der zur Verfügung stehenden Hardwareumgebung ausreichen schnell.
2. Die Zeit, die das Tool zum Laden und Speichern von Projekten, Diagrammen oder Spezifikationen benötigt, erlaubt ein zügiges Arbeiten.
3. Suchergebnisse stehen, in Anbetracht des bereits erfassten Datenvolumens, rasch zur Verfügung.
4. Die automatische Codegenerierung erfolgt ausreichend schnell.

h) Adaptivität

1. Das Werkzeug kann dem Lernfortschritt der Anwender angepasst werden, z.B. zur Verfügungstellung unterschiedlich detaillierter Menüs (Ausblenden von komplexen Befehlen), Ein- und Ausschalten von Sicherheitsabfragen und Hinweisdialogen, Expert oder Apprentice-Modus etc.
2. Die Oberfläche lässt sich individuell gestalten (Symbolleisten, Funktionstastenbelegung, etc.) und kann für jeden Benutzer spezifisch eingestellt werden.
3. Die Struktur der Menüs kann vom Anwender verändert werden (neue Befehle einfügen, Befehle umbenennen, etc.).
4. Es gibt eine Möglichkeit, benutzerdefinierbare Befehlsfolge aufzuzeichnen und später abzurufen (Makros).
5. Zur Erstellung von Dokumenten gibt es eine Scriptsprache, mit der Ausgaben frei gestaltbar sind. Alle Modellelemente können dabei als Variablen verwendet werden.
6. Das Werkzeug wird im Sourcecode ausgeliefert, was ein Maximum an Anpassungsmöglichkeiten bietet.

10 Literaturverzeichnis

- Hitz, M., Kappel, G., Kapsammer, E., Retschitzegger, W., UML@Work, 3. Auflage, dpunkt.Verlag GmbH, Heidelberg 2005.
- Grünsteidl, O., Lehner, J., Untersuchung und Vergleich UML-basierter Entwicklungswerkzeuge, Diplomarbeit, Universität Wien, 2000.
- Kelter, U., Bewertung und Auswahl von UML-Werkzeugen, http://pi.informatik.uni-siegen.de/kelter/lehre/02s/lm/lm_bau_20010407_a5.pdf.gz (25.07.2007), 2001.
- Meyer, J., Comparison of UML Modelling Tools: Sparx Systems “Enterprise Architect 5.0” and IBM Rational “Rose 2003 (2003.06.13)”, Dunstan Thomas Consulting, www.dthomas.co.uk/DTALM/downloads/EA-Rose.pdf (25.07.2007), 2005.